Index des opérateurs

Conversion de format de fichiers

Conversion entre formats d'image (gif, bmp, tiff, png, jpeg, pandore, etc).

- pany2pan - Conversion d'une image d'un format connu (GIF, BMP, etc) en image Pandore.
- pbmp2pan - Conversion d'une image BMP en image Pandore.
- ppan2bmp - Conversion d'une image Pandore en image BMP.
- pjpeg2pan - Conversion d'une image JPEG en image Pandore.
- ppan2jpeg - Conversion d'une image Pandore en image JPEG.
- ppng2pan - Conversion d'une image PNG en image Pandore.
- ppan2png - Conversion d'une image Pandore en image PNF.
- pgif2pan - Conversion d'une image GIF en image Pandore.
- ppan2gif - Conversion d'une image Pandore en image GIF.
- ppan2ppm - Conversion d'une image Pandore en image PPM (ascii).
- pppm2pan - Conversion d'une image PPM, PGM ou PBM en image Pandore.
- ptiff2pan - Conversion d'une image TIFF en image Pandore.
- ppan2tiff - Conversion d'une image Pandore en image TIFF.
- pvff2pan - Conversion d'une image VFF en image Pandore.
- ppan2vff - Conversion d'une image Pandore en image VFF (Sunvision).
- ppan2pan - Conversion d'une image Pandore en image Pandore.
- pras2pan - Conversion d'une image Sun raster en image Pandore.
- praw2pan - Conversion d'une image raw image (binaire) en image Pandore.
- ppan2raw - Conversion d'une image Pandore rn image raw (binaire).
- ppan2ps - Conversion d'une image Pandore en fichier Encapsulted PostScript.
- ppan2txt - Conversion d'une image Pandore en fichier texte.
- ptxt2pan - Conversion d'un fichier texte en image Pandore.
- pyuv2pan - Conversion d'une séquence d'image YUV en image Pandore.
- pfits2pan - Conversion d'une image FITS (Flexible Image Transport System) en image Pandore.
- ppan2fits - Conversion d'une image Pandore en niveaux de gris en image FITS (Flexible Image Transport System).
- pparrec2pan - Conversion PAR/REC format image file (Philips Medical System) to Pandore image file.
- panalyze2pan - Conversion d'une image Pandore en image ANALYZE 7.5.
- ppan2analyze - Conversion d'une image ANALYZE 7.5 en une image Pandore.
- ppan2d23d - Conversion d'une série d'images Pandore 2D en une image Pandore 3D.
- ppan3d22d - Conversion d'une image Pandore 3D en une série d'images Pandore 2D Pandore.

Conversion de type Pandore

Conversions de format d'objet Pandore: image de float en image de char, image couleur en image de niveaux de gris, etc.

- parray2im - Conversion d’une matrice de pixels en image.
- pim2array - Conversion d’une image en matrice de pixels.
- prg2im - Conversion d’une carte de régions en image de long signé.
- prg2imc - Conversion d’une carte de régions en image couleur équivalente.
- pim2rg - Conversion d’une image de niveaux de gris en carte de régions.
- pgr2im - Conversion d’une graphe en image de floats.
- prg2gr - Conversion d’une carte de régions en graphe.
- pgr2rg - Conversion d’un graphe en carte de régions.
- pim2sf - Conversion d’une image en image de floats.
- pim2uc - Conversion d’une image en image de unsigned char.
- pim2sl - Conversion d’une image en image de signed long.
- pim2d23d - Conversion d’une image ou d’une carte de régions 2D en image ou carte de régions 3D à 1 plan.
- pim3d22d - Conversion d’une image ou d’une carte de régions 3D à 1 plan en image ou carte de régions 2D.
- pimc2img - Conversion d’une image couleur en image de niveaux de gris.
- pimc2imx - Création d’une image multispectrale en image couleur.
- pimg2imc - Conversion d’une image en niveaux de gris en image color.
- pimg2imx - Conversion d’une image en niveaux de gris en image multispectrale.
- pimgs2imc - Création d’une image couleur à partir de 3 images de niveaux de gris.
- pimgs2imx - Création dune image multispectrale à partir de plusieurs images de niveaux de gris.
- pimx2img - Conversion d’une bande d’une image multispectrale en image de niveaux de gris.
- pimx2imc - Conversion d’une image multispectral en image couleur.

Conversion d’espace couleur

Conversions d’espace couleur d’une image couleur: rgb en hsl, rgb en yuv, etc.

- phsi2rgb - Conversion d’une image HSI en image RGB.
- phsl2rgb - Conversion d’une image HSL en image RGB.
- phsv2rgb - Conversion d’une image HSV en image RGB.
- plab2lch - Conversion d’une image Lab en image LCH.
- pluv2lch - Conversion d’une image L*u*v en image LCH.
- prgb2pca - Conversion d’une image rgb en image de composantes principales.
- prgb2ast - Conversion d’une image rgb en image AST.
- prgb2cmyk - Conversion d’une image RGB en image de type Cyan-Magenta-Yellow-Key.
- pcmyk2rgb - Conversion d’une image couleur de type Cyan-Magenta-Yellow-Key en RGB.
- prgb2gray - Conversion d’une image RGB en image de niveaux de gris.
- prgb2hsi - Conversion d’une image RGB en image HSI.
- prgb2hsl - Conversion d’une image RGB en image HSL.
- prgb2hsv - Conversion d’une image RGB en image HSV.
- prgb2i1i2i3 - Conversion d’une image RGB en image (i1,i2,i3).
- prgb2wry - Conversion d’une image RGB en image (wb,rg,yb).
- prgb2xyz - Conversion d’une image RGB en image XYZ.
- prgb2rngnbn - Conversion d’une image RGB en image RGB normalisée.
- prgb2ycbcr - Conversion d’une image RGB en image YCbCr.
- prgb2ych1ch2 - Conversion d’une image RGB en image YCh1Ch2.

- prgb2yiq - Conversion d’une image RGB en image YIQ.
- prgb2yuv - Conversion d’une image RGB en image YUV.
- pyuv2rgb - Conversion d’une image YUV en image RGB.
- pxyz2lab - Conversion d’une image XYZ en imagex Lab.
- pxyz2luv - Conversion d’une image XYZ en image L*u*v.
- pxyz2rgb - Conversion d’une image XYZ en image RGB.
- pgray2bw - Conversion d’une image en niveaux de gris en image noir et blanc.
- pgray2falsecolor - Conversion d’une image en niveaux de gris en fausses couleurs.

Arithmétique

Opérations arithmétiques binaires et unaires de base sur les images, les cartes de régions et les graphes.

- padd - Addition de 2 images.
- pdif - Différence de 2 images. Différence symétrique entre 2 cartes de régions.
- psub - Soustraction de 2 images. Différence non symétrique entre 2 cartes de régions.
- pmult - Multiplication de 2 images.
- pdiv - Division de 2 images.
- pmean - Moyennage de 2 images.
- pblend - Pondération enre deux images ou graphes.
- ppow - Puissance nième d’une image.
- psqrt - Racine carrée d’une image.
- pexp - Exponentiel d’une image.
- plog - Logarithme népérien d’une image.
- pmax - Maximum entre pixels de 2 images.
- pmin - Minimum entre pixels de 2 images.
- pabs - Valeur absolue d’une image.
- pround - Arrondi d’une image de réels.
- pclipvalues - Ecrétage de valeurs de pixel.
- pnormalization - Normalisation d’une image.
- pconvolution - Convolution d’une image par un noyau.
- psetcst - Affectation d’une valeur à une image.
- pdivcst - Division d’une image ou d’une carte de régions par une constante.
- pmultcst - Multiplication d’une image par une constante.
- paddcst - Addition d’une image par une constante.
- paddval - Addition d’une image avec des constantes stockées dans une collection.
- psubval - Soustraction d’une image avec des constantes stockées dans une collection.
- pmultval - Multiplication d’une image par des constantes stockées dans une collection.
- pdivval - Division d’une image par des constantes stockées dans une collection.
- plipadd - Addition de 2 images selon le modèle LIP.
- plipsub - Soustraction de 2 images selon le modèle LIP.
- plipmultcst - Multiplication d’une image par une constante selon le modèle LIP.
- pintegralimage - Calcule l’image intégrale.
- psquareintegralimage - Calcule l’image intégrale carrée.

Logique

Opérations logiques de type binaire (bit à bit) et booléennes entre images et graphes et les opérations ensemblistes entre cartes de régions.

- pand - Et binaire entre 2 objets. Intersection entre 2 cartes de régions.
- por - Ou binaire entre 2 objets. Union entre 2 cartes de régions.
- pxor - Ou exclusif binaire entre 2 objets. Différence symétrique entre 2 cartes de régions.
- pinverse - Inversion logique des valeurs d'un objet. Inversion des numéros de labels d'une carte de régions.
- pnot - Négation logique des valeurs de pixel d'une image. Complémentaire d'une carte de régions.
- pmask - Masquage d'un objet par un autre.

Transformation géométrique

Transformations géométriques du contenu des images: rotation, projection, symétrie, zoom, etc.

- paddborder - Agrandissement d'une image en lui ajoutant un bord.
- pmaxprojection - Projection orthogonale sur un axe d'une image selon les maxima.
- pmeanprojection - Projection orthogonale sur un axe d'une image selon la moyenne.
- pextrude1d22d - Propagation d'une valeur le long d'un axe.
- pflip - Construction du symétrique d'une image.
- protation - Construction de la rotation de contenu d'une image selon un axe.
- ptranslation - Construction du translaté d'une image.
- pscrolling - Construction de l'enroulé d'une image.
- ptransposition - Construction du transposé d'une image selon un axe.
- presize - Ajustement de la taille d'une image à une nouvelle taille.
- prescale - Augmentation ou réduction de la taille d'une image, d'une carte de région ou d'un graphe.
- plinearrescale - Augmentation ou réduction de la taille d'une image par interpolation bilinéaire.
- pbicubicrescale - Augmentation ou réduction de la taille d'une image par interpolation bicubique.
- pbellcrescale - Retaille d'une image par l'algorithme de Bell.
- phermiterescale - Retaille d'une image par l'algorithme de Hermite.
- planczosrescale - Retaille d'une image par l'algorithme de Lanczos.
- pmitchellrescale - Retaille d'une image par l'algorithme de Mitchell.

Utilitaire image

Opérations diverses de manipulation des images: génération de bruit, extraction, insertion, etc.

- psetpixel - Affecte une valeur particulière à un pixel donné.
- ppixelvalue - Affichage de la valeur d'un pixel d'une image ou d'un noeud d'un graphe donné.
- paddnoise - Génération de bruit aléatoire sur une image.
- pcliparea - Sélection d'une zone d'image, de carte de région ou de graphe.
- pcopyborder - Copie le bord d'une image vers une autre.
- psetborder - Affectation d'une valeur sur le bord d'une image.

- pextractsubimage - Extraction d’une sous-image d’une image.
- pinsertsubimage - Insertion d’une image dans une autre image.
- pnewimage - Création d’une nouvelle image.
- pshapedesign - Création d’une image vierge ou contenant une forme synthétique prédéfinie.
- pmergeimages - Regroupement de 4 sous-images en une seule.
- psplitimage - Eclatement d’une image en 4 sous-images.
- pgraylevel2depth - Conversion d’une image 2D de niveaux de gris en image 3D de niveaux.
- pdepth2graylevel - Conversion d’une image 3D en image 2D de niveaux de gris.
- paddslice - Ajout d’un plan dans une image 3D.
- premoveslice - Suppression d’un plan dans une image 3D.
- pgetband - Récupération une bande dans une image multispectrale (ou couleur).
- psetband - Remplacer une bande d’une image multispectrale (ou couleur).
- pgetslice - Récupération d’un plan dans une image 3D.
- psetlice - Remplacement d’un plan dans une image 3D.
- pgetwindowaroundpoints - Extraction des pixels dans la fenêtres autour de points spécifiés.

Transformation de lut

Opérations de transformation de la table des couleurs pour l’amélioration d’images.

- pextremumsharpening - Rehaussement du contraste par utilisation des valeurs extrémales.
- plineartransform - Transformation linéaire des niveaux de gris.
- plogtransform - Transformations des niveaux de gris par loi logarithmique ou exponentielle.
- ppowerlawtransform - Transformation des niveaux de gris par une loi de puissance.
- phistogramequalization - Rehaussement de contraste par égalisation d’histogramme.
- phistogramspecification - Rehaussement de contraste par spécification d’histogramme.

Filtrage spatial

Opérations de filtrage spatial linéaire et non linéaire.

- pvariancefiltering - Filtrage d’une image par variance.
- pmedianfiltering - Lissage d’une image par médian standard séparable.
- pmeanfiltering - Lissage d’une image par un filtre moyenneur linéaire.
- pnonlocalmedianfiltering - Filtrage médian non local.
- pnonlocalmeanfiltering - Filtrage moyenneur non local.
- pgaussianfiltering - Lissage d’une image par une gaussienne.
- pexponentialfiltering - Lissage par une exponentielle symétrique.
- padaptivemeanfiltering - Lissage d’une image préservant les contours.
- pderichesmoothing - Lissage de Deriche.
- pshensmoothing - Lissage de Shen-Castan.
- pmalikperonafiltering - Lissage d’une image par diffusion non linéaire selon l’algorithme de Malik-Peronna.
- pmcmfiltering - Lissage par diffusion par courbure moyenne.
- pnagaofiltering - Lissage par maximum d’homogénéité selon le masque de Nagao.
- poutrangefiltering - Lissage par filtre adaptatif basé sur le choix des voisins.
- psigmafiltering - Lissage par filtre adaptatif basé sur le choix des voisins.

- psnnfiltering - Lissage par filtre adaptatif : Symetric Nearest Neighbourghood.
- ppeergroupfiltering - Lissage d'une image couleur par Peer Group Filtering.
- pdenoisePDE - Régularisation d'images multivaluées par lissage anisotrope basé EDP.
- psharp - Rehaussement du contraste par convolution.

Approximation de surface

Opérations d'approximation du contenu d'une image par une surface.

- plinearregression - Calcul de l'approximation du fond d'une image en utilisant la regression linéaire.
- ppolynomialfitting - Calcul de l'approximation du fond d'une image en utilisant une approximation polynomiale.
- plegendrepolynomialfitting - Calcul de l'approximation du fond d'une image en utilisant une approximation par polynômes de Legendre.

Interpolation

Opérations d'interpolation de pixels manquants.

- plinearinterpolation - Remplacement de pixels manquants par interpolation linéaire des voisins.

Domaine fréquentiel

Les opérateurs fréquentiel permettent de passer du domaine spatial au domaine fréquentiel et vice et versa et de manipuler le contenu sous sa forme fréquentielle. Complex images are represented by two images.

- pfft - Calcul de la Transformée de Fourier Rapide d'une image.
- pifft - Transformée de Fourier Rapide Inverse d'une image.
- pbutterworthfilter - Génère un filtre passe-bas, passe-haut, coupe-bande ou passe-bande de Butterworth.
- pgaussianfilter - Génère un filtre Gaussien passe-bas, passe-haut, coupe-bande ou passe-bande.
- pmodulus - Calcul du module entre deux images.
- pphase - Calcul de la phase entre deux images.
- pfftconvolution - Convolution d'une image par un noyau.
- pfftdeconvolution - Déconvolution d'une image par un noyau.
- pfftcorrelation - Corrélation entre deux images.
- pfftshift - Permutation des 4 sous-images de la transformée de Fourier.
- pqmf - Génération d'un filtre QMF pour la transformée en ondelette.
- pdwt - Calcul de la transformée en ondelettes dyadiques biorthogonales d'une image.
- pidwt - Reconstruction d'une image décomposée en ondelettes dyadiques biorthogonales.
- psetsubband - Insertion d'une sous-bande dans une image de DWT.
- pgetsubband - Extraction d'une sous-bande d'une image de DWT.

Morphologie mathématique

Les opérateurs morphologiques sont des opérateurs variés basés sur l'utilisation d'un élément structurant et des opérations ensemblistes de type érosion/dilatation.

- pdilation - Dilatation des points de plus fort contraste d'une image.
- perosion - Erosion des points de plus fort contraste d'une image.
- pnonlocaldilation - Dilatation par régularisation non locale du laplacien.
- pnonlocalerosion - Erosion par régularisation non locale du laplacien.
- psedesign - Génération d'un élément structurant prédéfini.
- psedilation - Dilatation des points de fort contraste d'une image à partir d'un élément structurant donné.
- pseerosion - Erosion des points de fort contraste d'une image à partir d'un élément structurant donné.
- plineardilation - Dilatation des points de plus fort contraste d'une image avec un élément structurant linéaire.
- plinearerosion - Erosion des points de plus fort contraste d'une image avec un élément structurant linéaire.
- pgeodesicdilation - Dilatation géodésique des points de plus fort contraste de l'image.
- pgeodesicerosion - Erosion géodésique des points de plus fort contraste de l'image.
- pdilationreconstruction - Reconstruction morphologique par dilatation.
- perosionreconstruction - Reconstruction morphologique par érosion.
- pareaopening - Ouverture aérolaire (tueur de surface claire).
- pareaclosing - Fermeture aérolaire (tueur de surface sombre).
- pwatershed - Ligne de partage des eaux.
- phitormiss - Transformation de type tout ou rien.
- pskeletonization - Squelettisation d'objets binaires 2D.
- phomotopicskeletonization - Squelettisation homotopique d'objets binaires 3D.

Détection des points d'intérêt

Détection de points d'intérêt points de jonctions ou coins.

- pharris - Détection de points d'intérêt selon l'algorithme de Harris-Stephens.
- psusan - Détection de points d'intérêt selon l'algorithme SUSAN.

Détection de contours

Détection et localisation de contours dans les images.

- pgradneumann - Calcul du gradient d'une image par différences finies décentrées à droite avec conditions aux bords de Neumann.
- pdivneumann - Calcul de la divergence par différence finies décentré à gauche.
- pgradient - Calcul du module et de la direction du gradient par convolution.
- plaplacian - calcul du Laplacien d'une image par convolution.
- pprewitt - Module du gradient de Prewitt.
- proberts - Module du gradient de Roberts.
- psobel - Module du gradient de Sobel.
- pderiche - Détection et localisation des contours de Deriche.
- pshen - Détection et localisation des contours de Shen-Castan.

- pgradientthreshold - Estimation du bruit dans une image d'amplitude du gradient.
- pnonmaximasuppression - Suppression des points non maxima dans une image d'amplitude de gradient.
- pzerocross - Localisation des changements de signe des valeurs de pixels.

Traitement de contours

Les contours sont des chaînes de pixels non nuls reposant sur un fond nul. Ils sont connectés les uns aux autres selon la 8 (en 2D) ou la 26 (en 3D) connexité. Les images de contours sont des images de uchar (Img2duc).

- pdistance - Calcul d'une image de distance euclidienne aux contours.
- pdistance1 - Calcul d'une image de distance quelconque aux contours.
- pcontourextensionrect - Extension des points terminaux dans la direction du contour.
- pcontourextensionconic - Extension des points terminaux dans la direction du contour par une forme conique.
- pblindedgeclosing - Fermeture de contours par poursuite de contours.
- pedgedirection - Calcul la direction des contours.
- pedgeclosing - Fermeture de contours par poursuite du gradient.
- phoughlines - Détection et localisation des segments de droite dans une image de contours par la transformée de Hough.
- ppostthinning - Suppression des points de contours qui ne garantissent pas la 8 connexité ou 26 connexité).
- pelliptisoidalapproximation - Approximation ellipsoïdale d'un ensemble de points ou des contours d'une image.
- ppolygonalapproximation - Approximation polygonale des contours d'une image.
- pbarbremoval - Suppression des barbules sur leur longueur.
- pcontourselection - Sélection des chaînes de contours isolées sur leur longueur.
- pclosedcontourselection - Sélection des chaînes de contours fermées sur leur longueur.
- popencontourselection - Sélection de chaînes de contours ouvertes sur leur longueur.

Seuillage

Segmentation d'image par classification des pixels.

- pbinarization - Seuillage binaire d'une image à partir d'une valeur de seuil.
- padaptivemeanbinarization - Binarisation de l'image par adaptation locale basée sur la moyenne.
- pcorrelationbinarization - Binarisation de l'image par maximisation de la corrélation interclasse.
- pentropybinarization - Binarisation de l'image par maximisation de l'entropie interclasse.
- pniblackbinarization - Binarisation de l'image basée sur le contraste local selon la méthode de W. Niblack améliorée par J. Sauvola.
- pvariancebinarization - Binarisation de l'image par analyse de la variance interclasse selon l'algorithme de Otsu.
- pmassbinarization - Binarisation d'une image basé sur le pourcentage de niveaux de gris.
- pcontrastbinarization - Binarisation d'une image par analyse du contraste aux frontières.
- pthresholding - Seuillage d'une image, une carte de régions ou un graphe à partir d'une valeur de seuil.
- pmassthresholding - Seuillage d'une image de niveaux de gris basé sur le pourcentage de niveaux

de gris.
- pcontrastthresholding - Multi-seuillage de l'image par analyse du contraste aux frontières.
- phistothresholding - Multi-seuillage d'une image de niveaux de gris par une ligne de partage des eaux de son histogramme.
- pfuzzyclustering - Multi-seuillage d'une image par la méthode des k moyennes floues.
- pentropythresholding - Multi-seuillage d'une image par analyse de l'entropie des régions.
- pchanda - Multi-seuillage d'une image par analyse de la matrice de co-occurences selon Chanda.
- pderavi - Multi-seuillage d'une mage par analyse de la matrice de co-occurences selon Deravi.
- pfisher - Multi-seuillage d'une image par partionnement de l'histogramme des niveaux de gris.
- pweszka - Multi-seuillage d'une image par analyse de la matrice de co-occurences selon Weszka.

Segmentation

Opérations de segmentation d'images de pixels en carte de régions.

- pboundarylabeling - Etiquetage en régions d'une image de contours fermés.
- plabeling - Etiquetage des régions homogènes d'une image.
- pcontrastquadtree - Segmentation d'une image par quadtree (octree) selon le contraste.
- pcontrast1quadtree - Segmentation d'une image par quadtree (octree) selon le contraste
- pedgebasedragpruning - Séparation dans le graphe d'adjacence de régions séparées par un point de contour.
- pentropyquadtree - Segmentation d'une image par quadtree (octree) selon l'entropie.
- puniformityquadtree - Segmentation d'une image par quadtree (ou octree) selon l'uniformité.
- pvariancequadtree - Segmentation d'une image par quadtree selon la variance.
- pcontrastaggregation - Croissance des régions d'une carte selon le contraste intérieur.
- pmeanaggregation - Croissance des régions d'une carte selon la moyenne intérieure.
- pgaussaggregation - Croissance des régions d'une carte selon une distribution gaussienne.
- pvarianceaggregation - Croissance des régions d'une carte selon la variance intérieure.
- pcontrastmerging - Fusion prioritaire de régions selon le critère du contraste.
- pentropymerging - Fusion prioritaire de régions selon le critère de l'entropie.
- pmeanmerging - Fusion prioritaire de régions selon la différence de moyennes intérieures.
- pmumfordshahmerging - Fusion prioritaire de régions selon la variation d'énergie de Mumford Shah.
- puniformitymerging - Fusion prioritaire de régions selon le critère d'uniformité.
- pvariancemerging - Fusion prioritaire de régions selon le critère de la variance.
- pboundarymerging - Fusion prioritaire de régions selon le contraste aux frontières.
- phistomerging - Fusion prioritaire de régions selon la corrélation d'histogramme.
- plabelmerging - Fusion nominative de 2 régions.
- pinnermerging - Fusion de régions englobées dans d'autres régions.
- pinnermostmerging - Fusion de régions avec la région voisine la plus englobante.
- pvoronoi - Calcul de la partition de Voronoï.
- pcolorquantization - Réduction du nombre de couleurs utilisées pour coder une image.
- pgraphbasedsegmentation - Segmentation d'images couleur par l'analyse des frontières des régions.
- pmeanshiftsegmentation - Classification des pixels d'une image par l'algorithme Mean-Shift.
- psimplelineariterativeclustering - Segmentation d'une image couleur en superpixels.
- pseedplacement - Placement de germes de régions sur une grille régulière.

- psuperpixelsegmentation - Segmentation d’une image couleur en superpixels.

Traitement de région

Opérations sur carte de régions telles que la sélection de région ou l’étiquetage.

- pboundaryregularization - Régularisation des frontières des régions.
- pfillhole - Bouchage des trous dans les régions.
- pholeselection - Sélection des trous dans les régions.
- pconvexhull - Calcul de l’enveloppe convexe des régions.
- pboundary - Localisation des points de frontière des régions.
- pboundingbox - Calcul du rectangle exinscrit des régions.
- pcenterofmass - Localisation des centres de gravité de régions.
- plabelselection - Sélection d’une région par son numéro de label.
- plabelsselection - Sélection des régions spécifiées par une autre carte de régions.
- pinnerselection - Sélection des régions englobées dans une autre région.
- poutborderselection - Sélection des régions qui ne touchent pas le bord de l’image.
- plocationselection - Sélection des régions à partir de leur position.
- penergyselection - Sélection de régions sur leur valeur d’énergie intérieure.
- pmaximumselection - Sélection de régions sur leur valeur de maximum intérieur.
- pminimumselection - Sélection de régions sur leur valeur de minimum intérieur.
- pmeanselection - Sélection de régions sur leur valeur de moyenne intérieure.
- pvarianceselection - Sélection de régions sur leur valeur de variance.
- pcompactnessselection - Sélection de régions sur leur valeur de compacité.
- pconvexityselection - Sélection de régions sur leur valeur de convexité.
- pdensityselection - Sélection de régions sur le facteur de densité.
- pelongationselection - Sélection de régions sur leur valeur d’élongation.
- peulernumberselection - Sélection de régions sur leur valeur de nombre d’Euler.
- peccentricityselection - Sélection de régions sur leur valeur d’excentricité.
- porientationselection - Sélection de régions sur leur valeur d’orientation.
- pperimeterselection - Sélection de régions sur leur valeur de périmètre.
- prectangularityselection - Sélection de régions sur leur valeur de rectangularité.
- psphericityselection - Sélection de régions sur leur valeur de sphéricité.
- psizeselection - Sélection de régions sur leur valeur de taille.
- pareaselection - Sélection de régions sur leur valeur de surface.
- pvolumeselection - Sélection de régions sur leur valeur de volume.
- prelabelingfromarray - Relabelisation d’une carte de régions à partir des valeurs d’un vecteur d’etiquettes.
- prelabelingwithgraph - Renumérotation des régions d’une carte et des sommets du graphe associé.

Extraction de caractéristiques région

Extrait des mesures de caractéristiques topologique, géométrique et photométrique des régions d’une carte donnée.

- pareadisorderfactor - Calcul du désordre surfacique d'une carte de régions.
- pregioncompactness - Calcul de la compacité des régions.
- pregionconvexity - Calcul de la convexité des régions.
- pregiondensity - Calcul de la densité des régions.
- pregionelongation - Calcul de l'élongation des régions.
- pregionenergy - Calcul de l'énergie des régions.
- pregioneulernumber - Calcul du nombre d'Euler des régions.
- pregioneccentricity - Calcul de l'excentricité des régions.
- pregionmaximum - Calcul de la valeur maximale des régions.
- pregionminimum - Calcul de la valeur minimale des régions.
- pregionmean - Calcul de la moyenne des régions.
- pregionorientation - Calcul de l'orientation des régions.
- pregionperimeter - Calcul du périmètre des régions.
- pregionrectangularity - Calcul de la rectangularité des régions.
- pregionsphericity - Calcul de la sphéricité des régions.
- pregionvariance - Calcul de la variance des régions.
- pregionarea - Calcul de la surface des régions.
- pregionvolume - Calcul du volume des régions.
- pregionwidth - Calcul de la largeur des régions.
- pregionheight - Calcul de la hauteur des régions.
- pregiondepth - Calcul de la profondeur des régions.

Extraction de caractéristiques image

Extraction de caractéristiques statistiques sur ou entre images.

- plocalextrema - Localisation des points constituant un extréma dans au moins direction.
- plocalmaxima - Localisation des points constituant un maximum local.
- plocalminima - Localisation des points constituant un minimum local.
- pregionalminima - Localisation des points constituant un minimum régional.
- pregionalmaxima - Localisation des points constituant un maximum régional.
- pcontrastvalue - Calcul du contraste global d'une image ou d'un graphe.
- pcontrast1value - Calcul du contraste global d'une image ou d'un graphe.
- penergyvalue - Calcul de l'énergie d'une image (d'un graphe ou d'une carte de régions).
- pentropyvalue - Calcul de l'entropie d'une image (d'un graphe ou d'une carte de régions).
- psumvalue - Calcul de la somme des valeurs de pixels (ou de sommets).
- pvariancevalue - Calcul de la variance des valeurs de pixels (ou de sommets).
- pmaximumvalue - Recherche la valeur de pixel maximum dans une image (un graphe ou une carte de régions).
- pminimumvalue - Recherche la valeur de pixel minimum dans une image (un graphe ou une carte de régions).
- pmeanvalue - Calcul du niveaux de gris moyen d'une image (un graphe ou une carte de régions).
- pmedianvalue - Recherche de la valeur médiane d'une image.
- pmodevalue - Retourne la valeur de pixel la plus nombreuse dans une image (un graphe ou une carte de régions).
- pvaluenumber - Comptage du nombre de pixels (ou de sommets) non nuls dans une image.
- pvalueclassnumber - Comptage du nombre de valeurs différentes dans une image ou un graphe.

- pvaluerank - Détermination de la ième valeur d’une image.
- phistogram - Création d’un histogramme à partir d’une image.

Evaluation

Mesures de la qualité d’un traitement d’images avec ou sans référence.

- passessdetectionaccuracy - Évaluation de la précision de la détection des régions d’un résultat de segmentation par comparaison avec une segmentation de référence.
- passessfragmentationconsistency - Évaluation de la cohérence de la fragmentation des régions d’un résultat de segmentation par comparaison avec une segmentation de référence.
- passessboundaryprecision - Évaluation de la précision de la localisation des frontières des régions d’un résultat de segmentation par comparaison avec une segmentation de référence.
- passessshapefidelity - Évaluation de la fidélité de la forme des régions d’un résultat de segmentation par comparaison avec une segmentation de référence.
- passesstopologypreservation - Évaluation de la préservation de la topologie des régions d’un résultat de segmentation par comparaison avec une segmentation de référence.
- passesssegmentationalgorithm - Évaluation des performances d’un algorithme de segmentation basée sur des mesures de dissimilarité entre des résultats de segmentation et des segmentations de référence.
- pranksegmentationalgorithms - Classement d’algorithmes de segmentation à partir de mesures de dissimilarité entre des résultats de segmentation et des segmentations de référence.
- pranksegmentationalgorithmsfromfolders - Classement d’algorithmes de segmentation à partir de mesures de dissimilarité entre des résultats de segmentation et des segmentations de référence (complet).
- pdisplayperformancevalues - Affichage détaillé des erreurs de segmentation calculées par l’opérateur ’passesssegmentationalgorithm’.
- pvinet - Calcul de la mesure de dissimilarité entre 2 cartes de régions basé sur le nombre de pixels mal segmentés.
- pborsotti - Calcul du critère de qualité basé sur le nombre, l’aire et la variance des régions.
- pzeboudj - Calcul du critère de qualité basé sur la mesure de contraste inter et intra-regions.
- pinterregioncontrast - Calcul du critère de qualité basé sur une mesure d’uniformité inter-régions.
- pintraregionuniformity - Calcul du critère de qualité basé sur une mesure d’uniformité intra-régions.
- pmse - Calcul de l’erreur quadratique moyenne.
- ppsnr - Calcul du rapport signal sur bruit en crêtes.
- psnr - Calcul du rapport signal sur bruit.

Mouvement

Opérations de reconstruction du mouvement à partir d’images ou de séquences d’images.

- pblockmatching - Estimation du mouvement entre deux images par mise en correspondance de blocs.
- pplotquiver - Dessin d’un champ de vecteurs 2D à partir d’une image 2D multi-spectrale à deux composantes.
- pregistrationPDE - Estimation du champ de déplacement entre deux images.

Reconstruction

Opération de reconstruction de surface et d'orientation.

- pgetquadrangle - Sélection du meilleur quadrilatère parmi un ensemble de lignes.
- pquadrangle2rectangle - Corrige la distortion géométrique du contenu d'une image pour passer de la représentation d'un quadrilatère à un rectangle.
- pskewanglecorrection - Détecte une déviation du contenu de l'image puis rectifie l'image.

Reconnaissance de Formes

Opérations de reconnaissance des formes.

- pcrosscorrelation - Correlation d'une image par un noyau.

Collection

Les opérateurs de collection manipulent la composition des collections. Une collection regroupe des éléments hétérogènes (e.g., valeur, tableau, image, graphe) dans une structure unique.

- pcolcatenateitem - Concaténation de 2 collections.
- pcolremoveitem - Suppression d'un dans une collection.
- pcolrenameitem - Changement de nom d'un d'une collection.
- pcolgetvalue - Extraction d'une valeur numérique dans une collection.
- pcolsetvalue - Ajout d'une valeur numérique dans une collection.
- pcolgetimages - Extraction des images d'une collection.
- pcolgetobject - Extraction d'un objet Pandore à partir d'une collection.
- pcolsetobject - Ajout d'un objet Pandore dans une collection.
- pobject2col - Création d'une collection à partir d'un objet Pandore.
- pcol2csv - Conversion d'une collection en fichier texte au format csv.
- pcol2txt - Conversion d'une collection en fichier texte.
- ptxt2col - Construction d'une collection à partir d'un fichier texte.

Vecteurs

Définit quelques opérateurs sur les vecteurs stockés dans les collections.

- pcreatearray - Création d'une collection contenant un vecteur vierge.
- parray2array - Conversion du type d'une vecteur dans une collection.
- parraygetvalue - Extraction de la valeur d'un d'un vecteur dans une collection.
- parraysize - Retourne la taille d'un vecteur dans une collection.
- parrayargmax - Extraction des valeurs maxima entre plusieurs tableaux.
- parraycovarmat - Calcul de la matrice de covariance associée à un ensemble d'éléments.
- parraymean - Calcul des moyennes des valeurs de vecteurs.
- parraymedian - Calcul de la valeur médiane des valeurs de vecteurs.
- parraymode - Calcul de la valeur la plus fréquente dans un vecteur.
- parrayeuclideannorm - Calcul de la norme euclidienne de vecteurs.

- parraynorm - Normalisation des valeurs d'un vecteur entre 0 et 1.
- parraysnorm - Normalisation des valeurs de plusieurs vecteurs entre 0 et 1.
- parraysmin - Calcul des valeurs minimales de chaque tableau dans une collection.
- parraysmax - Calcul des valeurs maximales de chaque tableau dans une collection.
- parraysmean - Calcul des valeurs moyennes de chaque tableau dans une collection.
- pcorrelationcoefficient - Calcul du coefficient de correlation entre deux vecteurs.

Graphe

Opérations sur graphes. Un graphe est un ensemble de sommets relié par des arcs. Un noeud référence par indice un objet (par exemple une régions) dans un tableau d'objets.

- pbetagraph - Construction du béta-graphe d'un graphe.
- psig - Construction de la sphère d'influence d'un graphe.
- pmst - Construction de l'arbre de recouvrement minimal d'un graphe.
- pdelaunay - Construction du graphe de Delaunay discret.
- pgraphpruning - Suppression des arcs nuls et des sommets isolés.
- pedgecutting - Suppression des arêtes d'un graphe sur leur value.
- pleafcutting - Suppression des feuilles d'un graphe.
- pgraphneighbours - Valuation des sommets d'un graphe avec le nombre de sommets voisins.
- pgraphvisu - Visualisation des valeurs des sommets et des arcs d'un graphe.
- pedgevisu - Visualisation des poids des arêtes d'un graphe dans une image.
- pnodevisu - Visualisation des valeurs des sommets d'un graphe dans une image.
- pnodedisc - Visualisation des valeurs des noeuds d'un graphe.

Classification d'objets

Opération de classification d'objets. Les objets peuvent être n'importe quoi comme des pixels ou des régions.

- pgaussclassification - Classification selon un modèle gaussien.
- pkmeans - Classification automatique selon les K-moyennes.
- pknn - Classification selon les K plus proches voisins.

Visualisation

Utilitaires d'affichage graphique ou textuel du contenu des objets Pandore.

- pvisu - Affichage d'un fichier image de type Pandore.
- pdraw - Interface de dessin sur une image.
- pcontentsdisplay - Affichage du contenu d'un objet Pandore.
- pplot1d - Construction d'une image 2D couleur à partir d'une image 1D.
- pcolorcube - Visualisation de la récartition des couleurs d'une image dans un cube (représentant l'espace couleur).
- pcolorize - Colorisation des régions à partir de leur valeur moyenne.
- psuperimposition - Surimposition de contours sur une image.

Information

Utilitaires d’affichage des propriétés des objets Pandore.

- pproperty - Récupération de la valeur d’une propriété d’un objet Pandore.
- pfile - Présentation des caractéristiques d’un fichier Pandore.
- psetstatus - Affectation d’une valeur au statut courant.
- pstatus - Affichage de la valeur retournée par la dernière exécution d’un opérateur Pandore.
- pmanfr - Affichage en ligne de la documentation française associée à un opérateur.
- pversion - Affichage du numéro de version de la distribution Pandore.

Exotique

Opération sur images ne relevant pas directement du traitement d’images.

- prds - Construction d’un Random Dot Stereogram (stéréogramme sans motif).
- pstereogram - Construction d’une image de stéréogramme couleur.
- psubsampling - Sous-échantillonnage d’une image.

Équipe Image - Laboratoire GREYC
UMR CNRS 6072 - Ensicaen & Université de Caen, France
Last modified:

pabs

Valeur absolue d'une image ou d'un graphe.

Synopsis

pabs [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pabs** construit la valeur absolue d'une image. Chaque pixel de l'image de sortie *im_out* est construit avec la valeur absolue du pixel correspondant dans l'image d'entrée *im_in*.

```
if (pixel(im_in) < 0)
then pixel(im_out) = -pixel(im_in)
else pixel(im_out) = +pixel(im_in)
```

Pour les images non signées, l'image de sortie est identique à l'image d'entrée.

Pour une image couleur ou multispectrale, la valeur absolue est appliquée sur chacune des couleurs indépendamment.

Pour les graphes, le graphe de sortie est construit avec la valeur absolue des valeurs de noeuds.

L'image de sortie est du même type que l'image d'entrée.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule la différence entre les images a.pan et b.pan et stocke le résultat dans l'image d.pan:

```
psub a.pan b.pan c.pan
pabs c.pan d.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PAbs( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

padaptivemeanbinarization

Binarisation de l'image par adaptation locale basée sur la moyenne.

Synopsis

padaptivemeanbinarization *radius_x radius_y radius_z percent* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

La binarisation de Wellner est une binarisation adaptative qui opère sur une fenêtre glissante. Elle est basée sur l'analyse de la moyenne des pixels dans la fenêtre. Si la valeur du pixel central est inférieure à un certain pourcentage de la moyenne des valeurs des pixels dans la fenêtre alors le pixel devient noir sinon il devient blanc.

L'algorithme peut être résumé par :

```
im_out(x,y) = 0   si im_in(x,y) <= mean(fenetre(f(x,y), x,y, radius)) * (1-p/100)
            = 255 sinon
```

Paramètres

- *radius_x*, *radius_y*, *radius_z*: la demi taille de la fenêtre en x, y, et z. Une valeur typique pour radius_x est un 1/16 de la largeur de l'image.
- *percent* : le pourcentage de la moyenne des voisins. Une valeur typique est 15.

Entrées

- *im_in* : une image d'octets en niveaux de gris.

Sorties

- *im_out*: une image binaire.

Résultat

Retourne SUCCESS ou FAILURE.

Exemple

Binarisation de l’image page.pan :

```
padaptivemeanbinarization 16 16 0 15 examples/page.pan bin.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PAdaptiveMeanBinarization( const Img2duc &im_in, Img2duc
&im_out, int radius_x, int radius_y, int radius_z, const int
percent);
```

Références

B. Bradleya and G. Rothb, "Adaptive Thersholding using the Integral Image", Journal of Graphics, GPU, and Game Tools, 12(2), pp. 13-21, 2007.

Auteur: Sébastien Bernery

padaptivemeanfiltering

Lissage d'une image préservant les contours.

Synopsis

padaptivemeanfiltering *connexite* [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **padaptivemeanfiltering** lisse les régions homogènes de l'image *im_in1* tout en préservant les contours. *im_in2* est une image d'amplitude du gradient.

Le principe de l'algorithme consiste à remplacer chaque point de l'image par la moyenne des voisins du point voisin de ce point dont l'amplitude du gradient est minimale.

Il n'y a pas de lissage sur le bord d'épaisseur 2.

Paramètres

- *connexite* spécifie le nombre de voisins à prendre en compte dans le calcul de la moyenne de remplacement (4 ou 8).

Entrées

- *im_in1*: une image 2D.
- *im_in2*: une image 2D.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applique un filtre moyenneur adaptatif à l'image tangram.pan:

```
pgradient 1 tangram.pan a.pan b.pan
padaptivemeanfiltering 8 tangram.pan a.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PAdaptiveMeanFiltering( const Img2duc &im_in,Img2duc
&ima,Img2duc &im_out, Uchar connexite );
```

Auteur: Régis Clouard

padd

Addition de 2 images ou de 2 graphes.

Synopsis

padd [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **padd** calcule la somme des valeurs de niveaux de gris entre chaque pixel de l'image *im_in1* et de l'image *im_in2* correspondant.

Le résultat est mis dans l'image destination *im_out* dont le type dépend de celles des entrées en reprenant les mêmes conventions que le C.

Il n'y a pas de gestion du débordement de valeurs. La formule reprend exactement l'opérateur du C :

```
pixel(im_out) = pixel(im_in1) + pixel(im_in2);
```

Les deux images d'entrée *im_in1* ou *im_in2* doivent être de même type. Au besoin, il est nécessaire d'utiliser les opérateurs de coercition. Par contre, l'image de sortie est du type le plus grand possible par rapport au type des images d'entrée:

- Long entre images d'octets.
- Long entre images d'entiers.
- Float entre image de floats.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Entrées

- *im_in1*: une image ou un graphe.
- *im_in2*: une image ou un graphe.

Sorties

- *im_out*: une image ou un graphe. Le type de l'image dépend des types de l'image d'entrée:
    - Long si les entrées sont des images de Uchar.
    - Long si les entrées sont des images de Long.
    - Float si les entrées sont des images de Float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
padd a.pan b.pan result.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PAdd( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

paddborder

Agrandissement d'une image en lui ajoutant ou retirant un bord à 0.

Synopsis

paddborder *ll lr hu hu df db* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **paddborder** permet d'agrandir ou de réduire l'image *im_in* de *ll+lr* en largeur, de *hu+hd* en hauteur et de *df+db* en profondeur. L'image initiale *im_in* est centrée dans l'image de sortie *im_out*.

Si les valeurs des paramètres sont négatives cela correspond à une suppression du bord.

Il n'y a aucune interpolation de l'image initiale *im_in*. Elle conserve donc la même résolution dans l'image de sortie *im_out*.

Les valeurs de pixels ajoutées sont égales à 0.

Paramètres

- *df* donne la taille en pixels du bord devant pour les images 3D.
- *df* donne la taille en pixels du bord derrière pour les images 3D.
- *hu* donne la taille en pixels du bord en haut.
- *hd* donne la taille en pixels du bord en bas.
- *ll* donne la taille en pixels du bord à gauche.
- *lr* donne la taille en pixels du bord à droite.

  Les valeurs de profondeur *df* et *db* doivent être données pour une image 2D, mais elles sont ignorées.

  Si les valeurs des paramètres sont négatives cela correspond à une suppression du bord.

Entrées

- *im_in*: une image ou une carte de régions.

Sorties

- *im_out*: une objet de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Ajoute un bord vide autour de l'image tangram.pan.

```
paddborder 1 1 1 1 1 1 tangram.pan a.pan
```

Suppression d'un bord de l'image tangram.pan.

```
paddborder -1 -1 -1 -1 0 0 tangram.pan a.pan
```

Voir aussi

Transformation

Prototype C++

```
Errc PAddBorder( const Img2duc &im_in, Img2duc &im_out, int ll, int
lr, int hu, int hl );
```

Auteur: Régis Clouard

paddcst

Addition d'une constante aux valeurs d'une image, d'un graphe ou d'une carte de région.

Synopsis

paddcst *cst* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **paddcst** calcule l'image *im_out* par addition des valeurs de pixels de l'image *im_in* par la valeur *cst*.

Il y a écrétage du résultat si la valeur résultante est supérieure à la valeur maximale du type de l'image. La formule de calcul est la suivante :

```
pixel(im_out) = pixel(im_in) + cst;
```

Pour les images couleur et multispectrale, l'opérateur est appliqué sépérament sur chacune des bandes.

Pour une carte de régions, ce sont les valeurs des labels qui sont additionnées.

Pour un graphe, ce sont les valeurs des noeuds qui sont additionnées.

Paramètres

- *cst* est un réel.

Entrées

- *im_in*: une image, un graphe ou une carte de régions

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Pour les cartes de régions, retourne la valeur de label maximale.

Exemples

Ajoute 10 à chaque pixel de l’image tangram.pan:

```
paddcst 10 tangram.pan a.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PAddCst( const Img2duc &im_in, Img2duc &im_out, Uchar cst );
```

Auteur: Régis Clouard

paddnoise

Génération d'un bruit aléatoire sur une image.

Synopsis

paddnoise *loi moyenne ecart_type* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **paddnoise** permet d'ajouter du bruit artificiel à une image. Plusieurs lois de génération de bruit sont possibles (cf. le paramètre *loi*). L'image de sortie *im_out* est calculée selon la loi:

- Cas d'une loi additive : im_out = im_in + im_bruit;
- Cas d'une loi multiplicative : im_out = im_in * im_bruit;

où `im_bruit` est une image de bruit générée à partir de la loi choisie comme suit:

Soit u1 et u2 deux valeurs réelles aléatoires uniformément distribuée. sur l'intervalle [0..1], les valeurs de *bruit* pour les différentes lois sont calculées par:

- la loi gaussienne (algorithme de Box Muller):

```
z0=sqrt(-2.0*log(u1))*cos(2.0*M_PI*u2);
bruit[i] = ecart_type*z0 + moyenne;
```

- la loi exponentielle (méthode par inversion):

```
z0=-1.0*log(u1);
bruit[i]= ecart_type*z0 + moyenne;
```

- la loi uniforme (rem : ecart-type = (max-min)/sqrt(12)):

```
z0=(u1-0.5)*sqrt(12.0);
bruit[i] = ecart_type*z0 + moyenne;
```

- la loi triangulaire:

```
z0=(u1+u2-1.0)*sqrt(6.0);
bruit[i] = ecart_type*z0 + moyenne;
```

Paramètres

- *loi* indique la nature de bruit à ajouter parmi:
  - 1: bruit additif gaussien.
  - 2: bruit additif exponentiel.
  - 3: bruit additif uniforme.
  - 4: bruit additif triangulaire.

    - 11: bruit multiplicatif gaussien.
    - 12: bruit multiplicatif exponentiel.
    - 13: bruit multiplicatif uniforme.
    - 14: bruit multiplicatif triangulaire.
- La *moyenne* et l'*ecart-type* sont des réels, paramètres de la loi choisie.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE en cas de mauvais paramètrage.

Exemples

Ajoute un bruit gaussien avec un moyenne 0 et un écart type de 1,5 à l'image tangram.pan puis calcule le PSNR du filtre moyenneur:

```
paddnoise 1 0 1.5 tangram.pan a.pan
pmeanfilter 2 a.pan i1.pan
ppsnr 255 tangram.pan i1.pan
pstatus
```

Voir aussi

Utilitaire

Prototype C++

```
Errc PAddNoise( const Img2duc &im_in, Img2duc &im_out, int loi,
Float moyenne, Float ecart_type );
```

Auteur: Régis Clouard

paddslice

Ajout d'un plan dans une image 3D à partir d'une image 2D.

Synopsis

paddslice *direction* [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **paddslice** ajoute une image 2D à la fin ou au début d'une image 3D. La nouvelle image *im_out* a donc un plan de plus que l'image 3D d'entrée *im_in1*.

La première image 3D peut être construire à partir d'une image 2D avec l'opérateur **pim2d23d**.

Paramètres

- *direction* spécifie si l'image doit être ajoutée au début si *direction* < 0 ou à la fin si *direction* > 0 de l'image 3D.

Entrées

- *im_in1*: une image 3D.
- *im_in2*: une image 2D.

Sorties

- *im_out*: une image 3D du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Ajoute l'image a2d.pan à la fin de l'image a3d.pan:

```
paddslice 1 a3d.pan a2d.pan b3d.pan
```

Voir aussi

Utilitaire, pinsertslice, premoveslice, pim2d23d

Prototype C++

```
Errc PAddSlice( const Imx3d &im_in1, const Imx2d &im_in2, Imx3d
&im_out, int direction );
```

Auteur: Régis Clouard

paddval

Addition d'une image avec des constantes stockées dans une collection.

Synopsis

paddval [*-m mask*] [*col_in*|-] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **paddval** calcule l'image *im_out* par addition des valeurs de pixels de l'image *im_in* avec les valeur stockes dans la collection *col_in*. La première valeur de la collection est ajoutée à tous les pixels de la première bande, la seconde à tous les pixels de la seconde bande, etc.

Il y a écrétage du résultat si la valeur résultante est supérieure à la valeur maximale du type de l'image. La formule de calcul est la suivante :

```
val = pixel(im_in) + col_in;
if (val > MAX) pixel(im_out) = MAX;
else if (val < MIN) pixel(im_out) = MIN;
else pixel(im_out) = val;
```

Entrées

- *col_in*: une collection avec autant de valeurs réelles que de nombre de bandes pour l'image d'entrée (p. ex. 3 pour une image couleur).
- *im_in*: une image.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Ajoute a tangram.pan sa valeur moyenne:

```
pmeanvalue tangram.pan a.pan
paddval a.pan tangram.pan b.pan
```

Autres exemples

Voir aussi

Arithmetique

Prototype C++

```
Errc PAddVal( const Collection &col_in, const Img2duc &im_in,
Img2duc &im_out );
```

Auteur: Régis Clouard

panalyze2pan

Conversion d’une image au format ANALYZE 7.5 en une image Pandore.

Synopsis

panalyze2pan *im_in* [*im_out*|-]

Description

L’opérateur **panalyze2pan** convertit une image ANALYZE 7.5 en image Pandore.

Une image Analyze (7.5) est formée de deux fichiers dans le même dossier et avec le même nom de base:

- un fichier d’entête (suffixe .hdr). Il contient les informations sur le fichier image suivant, telles que la taille du volume, la taille d’un voxel.
- une fichier image (suffixe .img). Il contient les données de l’image.

L’image d’entrée *im_in* est l’un des deux fichiers ANALYZE. Le second fichier est alors lu en utilisant le même nom de base.

L’image finale *im_out* est toujours de type Imx3dsf.

Entrées

- *im_in*: l’un des deux fichiers ANALYZE 7.5 (soit .hdr .img).

Entrées

- *im_out*: une image Pandore (Imx3dsf).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l’image "brain" en image Pandore "a.pan" et affiche la bande #0:

```
panalyze2pan brain.hdr a.pan
pimx2img 0 a.pan | visu
```

Voir aussi

Conversion

Prototype C++

```
Errc PAnalyze2Pan( const char* filename,Pobject** obj_out );
```

Avertissement

Auteur: David Tschumperlé

pand

Et binaire entre images ou graphes et intersection entre cartes de régions.

Synopsis

pand [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pand** effectue le "et" bit à bit entre les deux images d'entrée *im_in1* et *im_in2*.

Pour les images de réelles, le "et" s'impante avec l'opérateur C '&' et s'applique sur chaque pixel :

```
pixel(im_out) = pixel(im_in1) & pixel(im_in2);
```

Pour les images réelles, le "et" est:

```
pixel(im_out) = pixel(im_in1) * pixel(im_in2);
```

Pour les images couleur et multispectrale, le "et" est calculé sur chacune des bandes séparément.

Pour les graphes, l'opérateur "et" s'implante par l'opérateur * entre les valeurs de noeud.

Pour les cartes de régions, le "et" correspond à l'intersection des régions. La carte de régions résultante *im_out* est composée des régions des deux cartes d'entrée occupant le même espace. Il n'est pas nécessaire qu'elles aient le même label.

Les deux entrées doivent être de même type.

Entrées

- *im_in1*: une image, un graphe ou une carte de régions.
- *im_in2*: une image, un graphe ou une carte de régions.

Sorties

- *im_out*: un objet du même type que im_in1 et im_in2.

Résultat

Retourne SUCCESS ou FAILURE.

Pour les cartes de région, retourne la valeur de label maximum.

Exemples

- Sélection des pixels des pièces de tangram :

```
pbinarization 100 1e30 examples/tangram.pan a.pan
pand examples/tangram.pan a.pan b.pan
```

Voir aussi

Logique

Prototype C++

```
Errc PAnd( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

pany2pan

Conversion d'une image d'un format connu en image Pandore.

Synopsis

pany2pan *im_in* [*im_out*|-]

Description

L'opérateur **pany2pan** permet de convertir une image d'un format connu en un fichier au format *Pandore*.

Les formats connus sont :

- BMP,
- JPEG,
- GIF,
- PNG,
- TIFF,
- PPM, PGM, PBM,
- PANDORE.

Entrées

- *im_in*: un fichier image.

Sorties

- *im_out*: une image Pandore.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pany2pan image.jpeg image.pan
```

Voir aussi

Conversion.

Auteur: Régis Clouard

pareaclosing

Fermeture aérolaire (tueur de surface sombre).

Synopsis

pareaclosing *connexity area* [*-m mask*][*im_in*|-][*im_out*|-]

Description

L'opérateur **pareaclosing** permet de supprimer les objets connexes sombres dont la surface, en nombre de pixels, est supérieure à la valeur du paramètre *area*.

L'algorithme présenté de façon naïve consiste à:

1. Pourcourir chaque niveau de gris de l'image d'entrée *im_in* par seuillage
2. et, pour chaque seuil, supprimer les régions binaires noires de surface inférieure au seuil *area*. Le résultat final est l'addition des résultats à chaque niveau de gris.

Paramètres

- *connexity* définit la relation de connexité entre pixels voisins : 4 ou 8 voisinage pour le 2D et 6 et 26 voisinage pour le 3D.
- *area* donne la taille maximale des surfaces à supprimer, en nombre de pixels.

Entrées

- *im_in*: une image 2D de niveaux de gris.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Fermeture des surfaces des pièces de tangram inférieures à 500 pixels.

```
pbinarization 90 1e30 examples/tangram.pan i1.pan
pareaclosing 8 500 i1.pan out.pan
```

Voir aussi

Morphologie, pareaopening

Prototype C++

```
Errc PAreaClosing( const Img2duc &im_in, Img2duc &im_out, int
connexity, int area );
```

Auteur: Régis Clouard

pareadisorderfactor

Calcul du désordre surfacique d'une carte de régions.

Synopsis

pareadisorderfactor [*-m mask*] [*rg_in*|-]

Description

L'opérateur **pareadisorderfactor** permet de calculer le paramètre de désordre de surface de la carte de régions *rg_in*.

Ce paramètre mesure l'homogénéité de la taille des régions. Il est calculé par la formule:

AD=1-1/(1+*écart_type(surfaces)/moyenne(surfaces)*).

Cette valeur est accessible par la commande **pstatus**.

Entrées

- *rg_in*: une carte de régions.

Résultat

Retourne la valeur du désordre qui appartient à l'intervalle [0..1].

Exemples

Affiche le facteur de désordre des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pareadisorderfactor b.pan
pstatus
```

Voir aussi

Caractérisation de région

Prototype C++

```
double PAreaDisorderFactor( const Reg2d &rg_in );
```

Auteur: François Angot

pareaopening

Ouverture aérolaire (tueur de surface claire).

Synopsis

pareaopening *connexity area* [*-m mask*][*im_in*|-][*im_out*|-]

Description

L'opérateur **pareaopening** permet de supprimer les objets connexes clairs dont la surface est supérieure à la valeur du paramètre *area*.

L'algorithme présenté de façon naïve consiste à:

1. Pourcourir chaque niveau de gris de l'image d'entrée *im_in* par seuillage,
2. et, pour chaque seuil, ne garder que des régions binaires blanches de surface inférieure au seuil *area*. Le résultat final est l'addition des résultats à chaque niveau de gris.

Paramètres

- *connexity* définit la relation de connexité entre pixels voisins : 4 ou 8 voisinage pour le 2D et 6 et 26 voisinage pour le 3D.
- *area* donne la taille maximale des surfaces à conserver.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Ouverture de surfaces inférieures à 500 pixels pour récupérer les pièces de tangram entières.

```
pbinarization 0 90 examples/tangram.pan i1.pan
pareaopening 8 500 i1.pan out.pan
```

Voir aussi

Morphologie, pareaclosing

Prototype C++

```
Errc PAreaOpening( const Img2duc &im_in, Img2duc &im_out, int
connexity, int area );
```

Auteur: Régis Clouard

pareaselection

Sélection de régions sur leur valeur de surface.

Synopsis

pareaselection *relation seuil* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pareaselection** permet de sélectionner les régions sur leur surface. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

La valeur de surface est calculée en nombre de pixels inclus dans la région et sur la frontière. Pour les creux entre deux pixels, on ajoute la moitié de la surface manquante. Sur l'exemple ci-dessous, la surface est de 10=8+4*0,5.

```
 xx
xxxx
 xx
```

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur entière en nombre de pixels.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *rg_out*: une carte de régions 2D.

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Exemples

Sélectionne les régions de surface = 50 pixels :

```
pareaselection 0 50  rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PAreaSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, Ushort seuil );
```

Auteur: Régis Clouard

parray2array

Conversion du type d'une vecteur dans une collection.

Synopsis

parray2array *attr type* [*col_in*|-] [*col_out*|-]

Description

L'opérateur **parray2array** crée une collection *col_out* contenant tous les attributs de *col_in* et dont le vecteur *attr* a été converti en un vecteur de type *type* (donnée. toutes lettres : `Char`, `Ushort`, ..., `Double`). La conversion est faite en reprenant la conversion du C (cast).

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Segmente l'image tangram.pan par classification des pixels selon les k-moyennes à partir des caractéristiques de moyenne et de variance :

```
pmeanfiltering 1 tangram.pan moy.pan
pvariancefilter 0 255 tangram.pan var.pan

pim2array data.1 moy.pan data1.colc
pim2array data.2 var.pan data2.colc
parray2array data.1 Float data1.colc data1.cold
parray2array data.2 Float data2.colc data2.cold
pcolcatenateitem data1.cold data2.cold data3.cold
parraysnorm data data3.cold data3.cold

pkmeans data attrib 5 100 data3.cold cluster.cold

pproperty 0 tangram.pan
w='pstatus'
pproperty 1 tangram.pan
```

```
h=‘pstatus‘

parray2im $h $w 0 attrib cluster.Cold kmeans.pan
pim2rg kmeans.pan classif1_out.pan
```

Voir aussi

Vecteur

Prototype C++

```
Errc PArray2Array( Collection &col_in_out, const std::string &attr,
const std::string &type );
```

Auteur: Alexandre Duret-Lutz

parray2im

Création d'une image à partir de vecteurs d'une collection.

Synopsis

parray2img *w h d name* [*col_in*|-] [*im_out*|-]

Description

L'opérateur **parray2img2d** crée une image *im_out* de taille *w*x*h*x*d* à partir des valeurs du tableau *name* de la collection *im_out*.

Si la collection contient 1 vecteur nommé *name*, alors **parray2im** construit une image en niveaux de gris du type des données du vecteur.

Si la collection contient 3 vecteurs nommés *name.1*, *name.2* et *name.3*, alors **parray2im** construit une image couleur du type des données du vecteur.

Paramètres

- *d* est le nombre de plans, *h* est le nombre de lignes, et *w* le nombre de colonnes de l'image résultante. Si la profondeur *d* est nulle alors l'image de sortie est une image 2D.
- *name* est le nom du vecteur des niveaux de gris ou le préfixe des noms des vecteurs de 3 composantes couleurs dans la collection.

Entrées

- *col_in*: une collection.

Sorties

- *im_out*: une image dont le type dépend des paramètres et des vecteurs.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit l'image 2D a.pan à partir du vecteur foo dans la collection col.pan

```
parray2im 256 256 0 foo col.pan a.pan
```

Voir aussi

Coercition, pim2array

Prototype C++

```
Errc PArray2Im( const std::string s, const Collection &c, Img2duc
&im_out );
```

Auteur: Alexandre Duret-Lutz

parrayargmax

Extraction des valeurs maxima entre plusieurs tableaux.

Synopsis

parrayargmax *attr_in attr_out* [*col_in*|-] [*col_out*|-]

Description

L'opérateur **parrayargmax** permet de construire le vecteur *attr_out* dans la collection de sortie *col_out* qui détient le numéro du vecteur qui détient la valeur maximale entre les valeurs de même rang des vecteurs *attr_in*.1 ... *attr_in*.n ... de *col_in*.

col_out contient le vecteur *attr_out* qui indique pour le rang i le numéro du tableau *attr-in*.i de *col_in* qui détient la ième valeur maximale.

Les vecteurs *attr_in*.1 .. *attr_in*.n et *attr_out* ont tous la même taille.

Paramètres

- *attr_in* est dérivée. `attr_in.1`, `attr_in.2`, ..., `attr_in.n` qui sont autant de tableaux recherchés dans *col_in*.
- *attr_out* est le nom d'un tableau de Ushorts, créé dans *col_out*, pour lequel le *i*ème indique le numéro du tableau contenant le *i*ème maximal.

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Voir aussi

Vecteur

Prototype C++

```
Errc PArrayArgMax( const std::string &attr_in, const Collection
&c_in, const std::string &attr_out, Collection &c_out );
```

Auteur: Alexandre Duret-Lutz

parraycovarmat

Calcul de la matrice de covariance associée à un ensemble d'éléments.

Synopsis

parraycovarmat *attr_in attr_out* [*col_in*|-] [*col_out*|-]

Description

L'opérateur **parraycovarmat** calcule la matrice de covariance A, son inverse A^{-1} , son déterminant, et le vecteur de moyennes des caractéristiques, à partir d'un ensemble de vecteurs de caractéristiques.

Paramètres

- la collection *col_in* contient les vecteurs de caractéristiques à partir desquels il faut faire les calculs. S'il y a n vecteurs de p caractéristiques chacun, la collection doit contenir p tableaux `attr_in.1`, `attr_in.2`, ..., `attr_in.p` de n flottants chacun.
- la collection *col_out* contient en sortie :
  - `attr_out.mat` : le tableau des `p*p` s de la matrice de covariance ;
  - `attr_out.inv` : le tableau des `p*p` s de l'inverse de la matrice de covariance ;
  - `attr_out.det` : le déterminant de la matrice de covariance
  - `attr_out.det` : le tableau des p s du vecteur de la moyenne des caractéristiques.

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Classification des bonbons de l'image jellybean.pan à partir d'exemples donnés dans le dossier 'base' (Unix version).

```
# Learning
classes=1
for i in base/*.pan
do
```

```
    pim2array ind $i /tmp/tmp1
    parray2array ind.1 Float /tmp/tmp1| parray2array ind.2 Float | parray2array ind.3 Float - a.pan
    parraycovarmat ind ind a.pan i-01.pan
    if [ -f base.pan ]
    then pcolcatenateitem i-01.pan base.pan base.pan
    else cp i-01.pan base.pan
    fi
    classes=‘expr $classe + 1‘
done
rm /tmp/tmp1

# Classification
pim2array ind jellybeans.pan a.pan
parray2array ind.1 Float a.pan| parray2array ind.2 Float | parray2array ind.3 Float - b.pan
pgaussclassification ind ind ind base.pan b.pan | parray2im $ncol $nrow 0 ind | pim2rg - out.pan
```

Voir aussi

Vecteur

Prototype C++

```
Errc PArrayCovarMat( const Collection &col_in, , Collection
&col_out, const std::string &attr_in, const std::string &attr_out );
```

Auteur: Alexandre Duret-Lutz

parrayeuclideannorm

Calcul de la norme euclidienne de vecteurs.

Synopsis

parrayeuclideannorm *in_attr out_attr* [*col_in*|-] [*col_out*|-]

Description

L'opérateur **parrayeuclideannorm** calcule la norme de vecteurs dans *col_in*. Ces vecteurs doivent être sauvés dans des tableaux nommés:

```
in_attr.1,
in_attr.2,
...
in_attr.n
```

Le résultat est un tableau appelé *out_attr* dans *col_out*.

Paramètres

- *in_attr* est le préfixe des noms d'attribut dans la collection d'entréee sur lesquels doit être calculé la norme.
- *out_attr* est le nom de l'attribut contenant la norme dans la collection de sortie.

Entrées

- *im_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Voir aussi

Vecteur

Prototype C++

```
Errc PArrayEuclideanNorm(const std::string &attr_in, const std::string &attr_out,
              const Collection &col_in, Collection &col_out);
```

Auteur: Alexandre Duret-Lutz

parraygetvalue

Extraction de la valeur d'un vecteur dans une collection.

Synopsis

parraygetvalue *num_elt nom_vecteur* [*col_in*|-]

Description

L'opérateur **parraygetvalue** permet de récupérer la valeur du *num_elt* ième élément du vecteur *nom_vecteur* dans la collection *col_in*.

La valeur de sortie est ensuite récupérable par l'opérateur **pstatus**.

Paramètres

- *nom_vecteur* est le nom du vecteur dans la collection.
- *num_elt* est le rang de l'élément dans le vecteur.

Entrées

- *col_in*: une collection.

Résultat

Retourne la valeur numérique de l'élément ou FAILURE si le nom du vecteur n'existe pas ou si le rang de l'élément n'est pas valide.

Exemples

Voir aussi

Vecteur

Prototype C++

```
Errc PArrayGetValue(Collection &c1,std::String vector_name, int
num_element);
```

Auteur: Régis Clouard

parraymean

Calcul des moyennes des valeurs de vecteurs.

Synopsis

parraymean *in_attr out_attr* [*col_in*|-] [*col_out*|-]

Description

L'opérateur **parraymean** permet de calculer les moyennes des valeurs du tableau *in_attr* dans la collection *col_in*.

La sortie est une valeur nommée *out_attr* dans le fichier *col_out*.

Paramètres

- *in_attr* est le nom de l'attribut dans la collection d'entréee sur lequel doit être calculé la moyenne.
- *out_attr* est le nom de l'attribut contenant la moyenne dans la collection de sortie.

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE

Exemples

Affiche la taille moyenne des pièces de tangram :

```
pbinarization 100 255 ~/pantheon/software/pandore/examples/tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionheight height b.pan c.pan
parraymean height mode c.pan d.pan
pcolgetvalue mode d.pan
pstatus
```

Voir aussi

Vecteur

Prototype C++

```
Errc PArrayMean( const std::string &a_in, const std::string &a_out,
                 const Collection &in, Collection &out);
```

Auteur: Alexandre Duret-Lutz

parraymedian

Calcul de la valeur médiane des valeurs de vecteurs.

Synopsis

parraymedian *in_attr out_attr* [*col_in*|-] [*col_out*|-]

Description

L'opérateur **parraymedian** permet de calculer la valeur médiane des valeurs du tableau *in_attr* dans la collection *col_in*.

La sortie est une valeur nommée *out_attr* dans le fichier *col_out*.

Paramètres

- *in_attr* est le nom de l'attribut dans la collection d'entréee sur lequel doit être calculé la valuer médiane.
- *out_attr* est le nom de l'attribut contenant la valeur médiane dans la collection de sortie.

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE

Exemples

Affiche la taille médiane des pièces de tangram :

```
pbinarization 100 255 ~/pantheon/software/pandore/examples/tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionheight height b.pan c.pan
parraymedian height median c.pan d.pan
pcolgetvalue median d.pan
pstatus
```

Voir aussi

Vecteur

Prototype C++

```
Errc PArrayMedian( const std::string &a_in, const std::string &a_out,
                   const Collection &in, Collection &out);
```

Auteur: Alexandre Duret-Lutz

parraymode

Calcul de la valeur la plus fréquente dans un vecteur.

Synopsis

parraymode *in_attr out_attr precision* [*col_in*|-] [*col_out*|-]

Description

L'opérateur **parraymode** permet de calculer la valeur la plus fréquente du tableau *in_attr* dans la collection *col_in*. Quand plusieurs modes existent c'est le plus petit qui est retourné.

La sortie est une valeur nommée *out_attr* dans le fichier *col_out*.

Paramètres

- *in_attr* est le nom de l'attribut dans la collection d'entréee sur lequel doit être calculé le mode.
- *out_attr* est le nom de l'attribut contenant la valeur du mode dans la collection de sortie.
- *precision* permet de spécifier le nombre de valeurs regroupées dans un même bin. Par exemple, si precision = 10 alors les valeurs de 0 à 9 sont regroupées dans un même bin, de même que les valeurs de 10 à 19, etc.

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE

Exemples

Affiche la taille la plus fréquente à des pièces de tangram une précison de 5 près :

```
pbinarization 100 255 ~/pantheon/software/pandore/examples/tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionheight height b.pan c.pan
parraymode height mode 5 c.pan d.pan
pcolgetvalue mode d.pan
pstatus
```

Voir aussi

Vecteur

Prototype C++

```
Errc PArrayMode( const std::string &a_in, const std::string &a_out,
                 const Collection &in, Collection &out,
                 int precision );
```

Auteur: Régis Clouard

parraynorm

Normalisation des valeurs d'un vecteur entre 0 et 1.

Synopsis

parraynorm *attr_in* [*col_in*|-] [*col_out*|-]

Description

La collection *col_out* est une copie de *col_in* dans laquelle le tableau *attr_in* a été converti en tableau de Doubles compris entre 0 et 1. Chaque nombre a été divisé par la valeur maximale de son type.

Paramètres

- *attr_in* est le nom du tableau à normaliser.

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Voir aussi

Vecteur

Prototype C++

```
Errc PArrayNorm( const std::string &attr_in, Collection &col_in );
```

Auteur: Alexandre Duret-Lutz

parraysize

Retourne la taille d'un vecteur dans une collection.

Synopsis

parraysize *attr_in* [*col_in*|-]

Description

Retourne la taille du vecteur *attr_in* dans la collection *col_in*. La valeur est accessible par la commande **pstatus**.

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne la taille du vecteur.

Exemples

Classification des bonbons de l'image jellybean.pan à partir d'exemples stockés dans le dossier 'base' (Unix version).

```
# Apprentissage
   classes=1;
   for i in base/*.pan
   do
      pim2array ind $i /tmp/tmp1
      parraysize ind.1 /tmp/tmp1
      size=‘pstatus‘
      pcreatearray ind.C Ushort $size $classes | pcolcatenateitem /tmp/tmp1 - i-01.pan
      if [ -f base.pan ]
      then pcolcatenateitem i-01.pan base.pan base.pan
      else cp i-01.pan base.pan
      fi
      classes=‘expr $classes + 1‘
   done

# Classification
   pproperty 0 jellybeans.pan
   ncol=‘pstatus‘
   pproperty 1 jellybeans.pan
   nrow=‘pstatus‘

   pim2array ind jellybeans.pan | pknn ind ind ind 10 base.pan - | parray2im $ncol $nrow 0 ind | pim2rg - out.pan
```

Voir aussi

Vecteur

Prototype C++

```
Long PArraySize( const Collection &col_in, const std::string
&attr_in );
```

Auteur: Alexandre Duret-Lutz

parraysmax

Calcul des valeurs maximales de chaque tableau dans une collection.

Synopsis

parrysmax *attr_in* [*col_in*|-] [*col_out*|-]

Description

La collection *col_out* est une construite avec les différentes valeurs maximales des tableaux laquelle les tableaux `attr_in.1`, `attr_in.2`, ..., `attr_in.n` de la collection d'entrée *col_in*.

Paramètres

- *attr_in* est le nom de base des tableaux à normaliser (attr_in.1, attr_in.2, ..).

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Retourne la collection avec les valeurs maximales de chacune des images de moyenne et de variance.

```
pmeanfiltering 1 tangram.pan moy.pan
pvariancefilter 0 255 tangram.pan var.pan

pim2array data.1 moy.pan data1.colc
pim2array data.2 var.pan data2.colc
parray2array data.1 Float data1.colc data1.cold
parray2array data.2 Float data2.colc data2.cold
pcolcatenateitem data1.cold data2.cold data3.cold
parraysmax data data3.cold data3.cold
```

Voir aussi

Vecteur

Prototype C++

```
Errc PArraysMax( Collection &col_in, Collection &col_out, const
std::string &attr_in );
```

Auteur: R&eacue;gis Clouard

parraysmean

Calcul des moyennes de chaque tableau dans une collection.

Synopsis

parrysmean *attr_in* [*col_in*|-] [*col_out*|-]

Description

La collection *col_out* est une construite avec les différentes valeurs moyennes des tableaux `attr_in.1`, `attr_in.2`, ..., `attr_in.n` de la collection d'entrée *col_in*.

Paramètres

- *attr_in* est le nom de base des tableaux à normaliser (attr_in.1, attr_in.2, ..).

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Retourne la collection avec les valeurs moyennes de chacune des images de moyenne et de variance.

```
pmeanfiltering 1 tangram.pan moy.pan
pvariancefilter 0 255 tangram.pan var.pan

pim2array data.1 moy.pan data1.colc
pim2array data.2 var.pan data2.colc
parray2array data.1 Float data1.colc data1.cold
parray2array data.2 Float data2.colc data2.cold
pcolcatenateitem data1.cold data2.cold data3.cold
parraysmean data data3.cold data3.cold
```

Voir aussi

Vecteur

Prototype C++

```
Errc PArraysMean( Collection &col_in, Collection &col_out, const
std::string &attr_in );
```

Auteur: R&eacue;gis Clouard

parraysmin

Calcul des valeurs minimales de chaque tableau dans une collection.

Synopsis

parrysmin *attr_in* [*col_in*|-] [*col_out*|-]

Description

La collection *col_out* est une construite avec les différentes valeurs minimales des tableaux `attr_in.1`, `attr_in.2`, ..., `attr_in.n` de la collection d'entrée *col_in*.

Paramètres

- *attr_in* est le nom de base des tableaux à normaliser (attr_in.1, attr_in.2, ..).

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Retourne la collection avec les valeurs minimales de chacune des images de moyenne et de variance.

```
pmeanfiltering 1 tangram.pan moy.pan
pvariancefilter 0 255 tangram.pan var.pan

pim2array data.1 moy.pan data1.colc
pim2array data.2 var.pan data2.colc
parray2array data.1 Float data1.colc data1.cold
parray2array data.2 Float data2.colc data2.cold
pcolcatenateitem data1.cold data2.cold data3.cold
parraysmin data data3.cold data3.cold
```

Voir aussi

Vecteur

Prototype C++

```
Errc PArraysMin( Collection &col_in, Collection &col_out, const
std::string &attr_in );
```

Auteur: R&eacue;gis Clouard

parraysnorm

Normalisation des valeurs de plusieurs vecteurs entre 0 et 1.

Synopsis

pnormnarrays *attr_in* [*col_in*|-] [*col_out*|-]

Description

La collection *col_out* est une copie de *col_in* dans laquelle les tableaux `attr_in.1`, `attr_in.2`, ..., `attr_in.n` ont été convertis en tableaux de Doubles compris entre 0 et 1. Chaque nombre a été divisé par la valeur maximale de son type (ce type doit être identique pour tous les tableaux).

Paramètres

- *attr_in* est le nom de base des tableaux à normaliser (attr_in.1, attr_in.2, ..).

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Segmente l'image tangram.pan par classification des pixels selon les k-moyennes à partoir des valeurs de moyenne et de variance :

```
pmeanfiltering 1 tangram.pan moy.pan
pvariancefilter 0 255 tangram.pan var.pan

pim2array data.1 moy.pan data1.colc
pim2array data.2 var.pan data2.colc
parray2array data.1 Float data1.colc data1.cold
parray2array data.2 Float data2.colc data2.cold
pcolcatenateitem data1.cold data2.cold data3.cold
parraysnorm data data3.cold data3.cold

pkmeans data attrib 5 100 data3.cold cluster.cold
```

```
pproperty 0 tangram.pan
w=‘pstatus‘
pproperty 1 tangram.pan
h=‘pstatus‘

parray2im $h $w 0 attrib cluster.Cold kmeans.pan
pim2rg kmeans.pan classif1_out.pan
```

Voir aussi

Vecteur

Prototype C++

```
Errc PArraysNorm( Collection &col_in_out, const std::string &attr_in
);
```

Auteur: Alexandre Duret-Lutz

passessboundaryprecision

Évaluation de la précision de la localisation des frontières des régions d'un résultat de segmentation par comparaison avec une segmentation de référence.

Synopsis

passessboundaryprecision *matching_algorithm_id matching_threshold* [*segmentation_result_in*|-] [*reference_segmentation_in*|-] [*col_out*|-]

Description

L'opérateur **passessboundaryprecision** permet de calculer 2 mesures pour évaluer la précision de la localisation des frontières des régions d'un résultat de segmentation par comparaison avec une segmentation de référence (vérité terrain).

La précision des frontières des régions est mesurée par deux erreurs :

- L'**erreur de déficit** de pixels qui rend compte de la proportion de pixels non détectés dans les régions détectées.
- L'**erreur d'excès** de pixels qui rend compte de la proportion de pixels erronés ajoutés aux régions détectées.

Les valeurs d'erreurs sont dans l'intervalle [0..1], où 0 signifie aucune erreur et 1 le pire résultat. Une erreur de déficit de *x* signifie que *x * 100* pourcent des pixels des régions ne sont pas détectés en moyenne par région. Uen erreur d'excès de *x* signifie que *x * 100* pourcent des pxiels des segments sont en dehors des frontières des régions correspodnantes en moyenne average par segment. Le résultat est stocké dans la collection de sortie *col_out* qui contient les valeurs pour chacune des 2 mesures d'erreur.

Les mesures sont calculées sur la base d'un graphe de correspondance entre les segments du résultat de la segmentation et les régions de la segmentation de référence. Deux types de mise en correspondance sont possibles selon le paramètre *matching_algorithm_id* : le premier autorise la sur-segmentation et la sous-segmentation et le second ne permet que la mise en correspondance unique, un segment avec une région. Dans ce graphe, un segment *S* détecte une région *R* si la surface de recouvrement | R * S | est telle que :

```
| R * S |                                    | R * S |
--------- >= matching_threshold and  --------- >= matching_threshold
  | R |                                      | S |
```

Paramètres

- *matching_algorithm_id* : spécifie le numéro de l'algorithme de mise en correspondance à utiliser :
  - 0 : pour une correspondance de type 1-n et n-1. Un segment du résultat de segmentation peut regrouper plusieurs régions de la référence (sous-segmentation), et une région de la référence peut être découpée en plusieurs segments du résultat de segmentation (sur-segmentation). Toutefois, un segment ou une région ne peuvent participer à la fois à une sur-segmentation et à une sous-segmentation.
  - 1 : pour une correspondance de type 1-1. Un segment de la segmentation ne peut être mis en correspondance qu'avec au plus une région de la référence, et une région de la référence ne peut être mise en correspondance qu'avec au plus un segment du résultat de la segmentation.
- *matching_threshold* : indique la proportion minimale de surface de recouvrement entre une région et un segment pour accepter une détection. C'est une valeur entre [0,1] où la valeur *x* correspond à un recouvrement minimum de (*x**100)%.

Entrées

- *segmentation_result_in* : une carte de régions contenant le résultat d'une segmentation.
- *reference_segmentation_in* : une carte de régions contenant la segmentation de référence.

Sorties

- *col_out* : une collection avec les 2 valeurs d'erreur.

Résultat

Retourne SUCCESS ou FAILURE en cas de problème.

Exemples

Évaluation de la précision de la localisation des frontières des régions d'un résultat de la segmentation des régions avec un taux de recouvrement minimum de 50% :

```
passessboundaryprecision 0 0.5 resultimages/algo001/tangram.pan groundtruths/expert001/tangram.pan errors.pan
pvisu errors.pan
```

Voir aussi

Evaluation, passessdetectionaccuracy, passessfragmentationconsistency, passessshapefidelity, passesstopologypreservation, passesssegmentationalgorithm, pranksegmentationalgorithms, pranksegmentationalgorithmsfromfolders

Prototype C++

```
Errc PAssessBoundaryPrecision( const Reg2d &segmentation_result_in,
const Reg2d &reference_segmentation_in, Collection &cold, const int
matching_algorithm_id, const float matching_threshold );
```

Auteur : Régis Clouard

passessdetectionaccuracy

Évaluation de la précision de la détection des régions d'un résultat de segmentation par comparaison avec une segmentation de référence.

Synopsis

passessdetectionaccuracy *matching_algorithm_id matching_threshold* [*segmentation_result_in*|-] [*reference_segmentation_in*|-] [*col_out*|-]

Description

L'opérateur **passessdetectionaccuracy** permet de calculer 2 mesures pour évaluer la précision de la détection des régions d'un résultat de segmentation par comparaison avec une segmentation de référence (vérité terrain).

La précision de la détection est mesurée par deux erreurs :

- L'**erreur de rappel** qui rend compte de la proportion de faux négatifs.
- L'**erreur de précision** qui rend compte de la proportion de faux positifs.

Les valeurs d'erreurs sont dans l'intervalle [0..1], où 0 signifie aucune erreur et 1 le pire résultat. Une erreur de rappel de *x* signifie que *x * 100* pourcent des régions de la référence ne sont pas détectées dans le résultat de segmnatation. Une erreur de précision de *x* signifie que *x * 100* pourcent des segments ne détectent aucune région. Le résultat est stocké dans la collection de sortie *col_out* qui contient les valeurs pour chacune des 2 mesures d'erreur.

Les mesures sont calculées sur la base d'un graphe de correspondance entre les segments du résultat de segmentation et les régions de la segmentation de référence. Deux types de mise en correspondance sont possibles selon le paramètre *matching_algorithm_id* : le premier autorise la sur-segmentation et la sous-segmentation et le second ne permet que la mise en correspondance unique, un segment avec une région. Dans ce graphe, un segment *S* détecte une région *R* si la surface de recouvrement | R * S | est telle que :

```
| R * S |                                    | R * S |
--------- >= matching_threshold and  --------- >= matching_threshold
  | R |                                      | S |
```

Paramètres

- *matching_algorithm_id* : spécifie le numéro de l'algorithme de mise en correspondance à utiliser :
  - 0 : pour une correspondance de type 1-n et n-1. Un segment du résultat de segmentation peut regrouper plusieurs régions de la référence (sous-segmentation), et une région de la référence peut être découpée en plusieurs segments du résultat de segmentation (sur-segmentation). Toutefois, un segment ou une région ne peuvent participer à la fois à une

sur-segmentation et à une sous-segmentation.
    - 1 : pour une correspondance de type 1-1. Un segment du résultat de la segmentation ne peut être mis en correspondance qu'avec au plus une région de la référence, et une région de la référence ne peut être mise en correspondance qu'avec au plus un segment du résultat de la segmentation.
- *matching_threshold* : indique la proportion minimale de surface de recouvrement entre une région et un segment pour accepter une détection. C'est une valeur entre [0,1] où la valeur *x* correspond à un recouvrement minimum de (*x**100)%.

Entrées

- *segmentation_result_in* : une carte de régions contenant le résultat d'une segmentation.
- *reference_segmentation_in* : une carte de régions contenant la segmentation de référence.

Sorties

- *col_out* : une collection avec les 2 valeurs d'erreur.

Résultat

Retourne SUCCESS ou FAILURE en cas de problème.

Exemples

Évaluation de la précision de la détection des régions d'un résultat de la segmentation avec un taux de recouvrement minimum de 50% :

```
passessdetectionaccuracy 0 0.5 resultimages/algo001/tangram.pan groundtruths/expert001/tangram.pan errors.pan
pvisu errors.pan
```

Voir aussi

Evaluation, passessfragmentationconsistency, passessboundaryprecision, passessshapefidelity, passesstopologypreservation, passesssegmentationalgorithm, pranksegmentationalgorithms, pranksegmentationalgorithmsfromfolders

Prototype C++

```
Errc PAssessDetectionAccuracy( const Reg2d &segmentation_result_in,
const Reg2d &reference_segmentation_in, Collection &cold, const int
matching_algorithm_id, const float matching_threshold );
```

Auteur : Régis Clouard

passessfragmentationconsistency

Évaluation de la cohérence de la fragmentation des régions d'un résultat de segmentation par comparaison avec une segmentation de référence.

Synopsis

passessfragmentationconsistency *matching_threshold* [*segmentation_result_in*|-] [*reference_segmentation_in*|-] [*col_out*|-]

Description

L'opérateur **passessfragmentationconsistency** permet de calculer 2 mesures pour évaluer la cohérence de la fragmentation des régions d'un résultat de segmentation par comparaison avec une segmentation de référence (vérité terrain).

La cohérence de la fragmentation est mesurée par deux erreurs :

- L'**erreur de sous-segmentation** qui rend compte de la proportion de régions agglomérées par segment.
- L'**erreur de sur-segmentation** qui rend compte de la proportion de fragmentation des régions en plusieurs segments.

Un segment *S* détecte une région *R* si la surface de recouvrement | R * S | est telle que :

```
| R * S |                                    | R * S |
--------- >= matching_threshold and  --------- >= matching_threshold
  | R |                                    | S |
```

Les valeurs d'erreurs sont des valeurs réelles et plus la valeur est élevée plus l'erreur est grande En fait, une erreur de sur-segmentation de *x* signifie que les régions sont découpées en moyenne en *2^x* segments. Une erreur de sous-segmentation de *x* signifie que les r&eacutegions; regroupent en moyenne en *2^x* segments. Le résultat est stocké dans la collection de sortie *col_out* qui contient les valeurs pour chacune des 2 mesures d'erreur.

Paramètres

- *matching_threshold* : indique la proportion minimale de surface de recouvrement entre une région et un segment pour accepter une détection. C'est une valeur entre [0,1] où la valeur *x* correspond à un recouvrement minimum de (*x**100)%.

Entrées

- *segmentation_result_in* : une carte de régions contenant le résultat d'une segmentation.
- *reference_segmentation_in* : une carte de régions contenant la segmentation de référence.

Sorties

- *col_out* : une collection avec les 2 valeurs d'erreur.

Résultat

Retourne SUCCESS ou FAILURE en cas de problème.

Exemples

Évaluation de la cohérence de la fragmentation des régions d'un résultat de la segmentation avec un taux de recouvrement minimum de 50% :

```
passessfragmentationconsistency 0.5 resultimages/algo001/tangram.pan groundtruths/expert001/tangram.pan errors.pan
pvisu errors.pan
```

Voir aussi

Evaluation, passessdetectionaccuracy, passessboundaryprecision, passessshapefidelity, passesstopologypreservation, passesssegmentationalgorithm, pranksegmentationalgorithms, pranksegmentationalgorithmsfromfolders

Prototype C++

```
Errc PAssessFragmentationConsistency( const Reg2d
&segmentation_result_in, const Reg2d &reference_segmentation_in,
Collection &cold, const float matching_threshold );
```

Auteur : Régis Clouard

passesssegmentationalgorithm

Évaluation des performances d'un algorithme de segmentation basée sur des mesures de dissimilarité entre des résultats de segmentation et des segmentations de référence.

Synopsis

passesssegmentationalgorithm [*-v*] *matching_algorithm_id matching_threshold segmentation_result_path reference_segmentation_path* [*col_out1*|-] [*col_out2*|-]

Description

L'opérateur **passesssegmentationalgorithm** calcule des mesures de dissimilarité entre des résultats de segmentation obtenus par un algorithme et des segmentations de référence faites sur les mêmes images. Cinq indicateurs de dissimilarité sont évalués, et à chaque fois, deux mesures sont calculées avec une valeur entre 0 et 1 :

- **Indicateur 1** : La **précision de la détection** : Les deux erreurs sont :
  - L'erreur de rappel qui rend compte de la proportion de faux négatifs.
  - L'erreur de précision qui rend compte de la proportion de faux positifs.
- **Indicateur 2** : La **cohérence de la fragmentation** : Les deux erreurs sont :
  - L'erreur de sous-segmentation qui rend compte de la proportion de régions agglomérées par segment.
  - L'erreur de sur-segmentation qui rend compte de la proportion de fragmentation des régions en plusieurs segments.
- **Indicateur 3** : La **localisation des frontières** : Les deux erreurs sont :
  - L'erreur de déficit de pixels qui rend compte de la proportion de pixels non détectés dans les régions détectées.
  - L'erreur d'excès de pixels qui rend compte de la proportion de pixels erronés ajoutés aux régions détectées.
- **Indicateur 4** : Le **respect de la forme** : Les deux erreurs sont :
  - L'erreur de forme due à l'omission de surface des régions.
  - L'erreur de forme due à l'ajout de surface aux régions.
- **Indicateur 5** : La **préservation de la topologie** : Les deux erreurs sont :
  - L'erreur d'ajout de trou qui rend compte de la proportion de faux trous détectés.
  - L'erreur de suppression de trou qui rend compte de la proportion de trous non détectés.

Les mesures sont calculées sur la base d'un graphe de correspondance entre les segments des résultats de segmentation et les régions des segmentations de référence. Deux types de mise en correspondance sont possibles selon le paramètre *matching_algorithm_id* : le premier autorise la sur-segmentation et la sous-segmentation et le second ne permet que la mise en correspondance unique. Dans ce graphe, un segment *S* détecte une région *R* si la surface de recouvrement | R * S | est telle que :

$$\frac{|R * S|}{|R|} \geq matching\_threshold \text{ and } \frac{|R * S|}{|S|} \geq matching\_threshold$$

Les résultats de segmentation et les segmentations de référence sont des cartes de régions.

Plusieurs segmentation de référence peuvent exister pour chaque image test. Le dossier *reference_segmentation_path* doit être organisé en sous-dossiers correspondant à chacune des expertises, par exemple *expert001*, *expert002*, etc.

Le dossier *segmentation_result_path* ainsi que chaque sous-dossier de *reference_segmentation_path* doivent être organisés de la même façon, avec les mêmes sous-dossiers et les mêmes noms d'image.

Le résultat des 10 mesures d'erreurs pour chaque résultat de segmentation est stocké dans la collection *col_out1* (sous la forme numérateur / dénominateur). Quand plusieurs segmentations de référence existent, c'est l'erreur minimale qui est gardée. La collection de sortie *col_out2* contient les valeurs d'erreur moyennes prenant en compte les résultats de segmentation.

Paramètres

- *-v* : mode verbeux
- *matching_algorithm_id* : spécifie le numéro de l'algorithme de mise en correspondance à utiliser :
  - 0 : pour une correspondance de type 1-n et n-1. Un segment d'un résultat de segmentation peut regrouper plusieurs régions de la référence (sous-segmentation), et une région de la référence peut être découper en plusieurs segments d'un résultat de segmentation (sur-segmentation). Toutefois, un segment ou une région ne peut participer à la fois à une sur-segmentation et à une sous-segmentation.
  - 1 : pour une correspondance de type 1-1. Un segment de la segmentation ne peut être mis en correspondance qu'avec au plus une région de la référence, et une région de la référence ne peut être mise en correspondance qu'avec au plus un segment de la segmentation.
- *matching_threshold* : indique la proportion minimale de surface de recouvrement entre une région et un segment pour accepter une détection. C'est une valeur entre [0,1] où la valeur *x* correspond à un recouvrement de (*x**100)%.
- *segmentation_result_path* : le chemin vers le dossier des résultats de segmentation de l'algorithme. Ce dossier peut être organisé en sous-dossiers.
- *reference_segmentation_path* : le chemin vers le dossier des segmentations de référence. Le dossier est divisé en autant de sous-dossiers qu'il y a d'expertises disponibles sur les images. Chaque sous-dossier d'expertise est organisé de la même façon qu'un sous-dossier du dossier *segmentation_result_path* avec les mêmes noms d'image.

Sorties

- *col_out1*: une collection avec les 10 valeurs d'erreur de segmentation pour chaque image test.
- *col_out2*: une collection avec les 10 valeurs moyenne d'erreur de segmentation qui résument les performances de l'algorithme.

Résultat

Retourne SUCCESS ou FAILURE en cas de problème.

Exemples

Évaluation de la qualité de l'algorithme 'algo001' à partir de ses résultats stockés dans le dossier 'images/resulimages/algo001:

```
passesssegmentationalgorithm 0 0.5 images/resultimages/algo001 images/groundtruths detail_errors.pan total_errors.pan
pdisplayperformancevalues detail_errors.pan total_errors.pan
```

Voir aussi

Evaluation, passessdetectionaccuracy, passessfragmentationconsistency, passessboundaryprecision, passessshapefidelity, passesstopologypreservation, pranksegmentationalgorithms, pranksegmentationalgorithmsfromfolders

Prototype C++

```
Errc PAssessSegmentationAlgorithm( int matching_algorithm_id, float
matching_threshold, std::string segmentation_result_path,
std::string reference_segmentation_path, Collection & col_out1,
Collection & col_out2 );
```

Auteur: Régis Clouard

passessshapefidelity

Évaluation de la fidélité de la forme des régions d'un résultat de segmentation par comparaison avec une segmentation de référence.

Synopsis

passessshapefidelity *matching_algorithm_id matching_threshold* [*segmentation_result_in*|-] [*reference_segmentation_in*|-] [*col_out*|-]

Description

L'opérateur **passessshapefidelity** permet de calculer 2 mesures pour évaluer la fidélité de la forme des régions d'un résultat de segmentation par comparaison avec une segmentation de référence (vérité terrain).

La fidélité de la forme est mesurée par deux erreurs :

- L'**erreur de forme due à l'omission de surface** des régions.
- L'**erreur de forme due à l'ajout de surface** aux régions.

Les valeurs d'erreurs sont des valeurs réelles. Une erreur de forme de commission de surface de valeur *d* signifie que les points de frontière de la surface en excès des segments est à une distance moyenne de *d* pixels d'au moins un point frontière de la région correspondante ou vice versa. Le résultat est stocké dans la collection de sortie *col_out* qui contient les valeurs pour chacune des 2 mesures d'erreur.

Les mesures sont calculées sur la base d'un graphe de correspondance entre les segments du résultat de la segmentation et les régions de la segmentation de référence. Deux types de mise en correspondance sont possibles selon le paramètre *matching_algorithm_id* : le premier autorise la sur-segmentation et la sous-segmentation et le second ne permet que la mise en correspondance unique, un segment avec une région. Dans ce graphe, un segment *S* détecte une région *R* si la surface de recouvrement | R * S | est telle que :

```
| R * S |                                    | R * S |
--------- >= matching_threshold and  --------- >= matching_threshold
  | R |                                      | S |
```

Paramètres

- *matching_algorithm_id* : spécifie le numéro de l'algorithme de mise en correspondance à utiliser :
  - 0 : pour une correspondance de type 1-n et n-1. Un segment du résultat de la segmentation peut regrouper plusieurs régions de la référence (sous-segmentation), et une région de la référence peut être découpée en plusieurs segments du résultat de la segmentation (sur-segmentation). Toutefois, un segment ou une région ne peuvent participer à la fois à une

sur-segmentation et à une sous-segmentation.
    - 1 : pour une correspondance de type 1-1. Un segment de la segmentation ne peut être mis en correspondance qu'avec au plus une région de la référence, et une région de la référence ne peut être mise en correspondance qu'avec au plus un segment du résultat de la segmentation.
- *matching_threshold* : indique la proportion minimale de surface de recouvrement entre une région et un segment pour accepter une détection. C'est une valeur entre [0,1] où la valeur *x* correspond à un recouvrement minimum de (*x**100)%.

Entrées

- *segmentation_result_in* : une carte de régions contenant le résultat d'une segmentation.
- *reference_segmentation_in* : une carte de régions contenant la segmentation de référence.

Sorties

- *col_out* : une collection avec les 2 valeurs d'erreur.

Résultat

Retourne SUCCESS ou FAILURE en cas de problème.

Exemples

Évaluation de la fidélité de la forme des régions du résultat de la segmentation avec un taux de recouvrement minimum de 50% :

```
passessshapefidelity 0 0.5 resultimages/algo001/tangram.pan groundtruths/expert001/tangram.pan errors.pan
pvisu errors.pan
```

Voir aussi

Evaluation, passessedetectionaccuracy, passessfragmentationconsistency, passessboundaryprecision, passesstopologypreservation, passesssegmentationalgorithm, pranksegmentationalgorithms, pranksegmentationalgorithmsfromfolders

Prototype C++

```
Errc PAssessshapeFidelity( const Reg2d &segmentation_result_in,
const Reg2d &reference_segmentation_in, Collection &cold, const int
matching_algorithm_id, const float matching_threshold );
```

Auteur : Régis Clouard

passesstopologypreservation

Évaluation de la préservation de la topologie des régions d'un résultat de segmentation par comparaison avec une segmentation de référence.

Synopsis

passesstopologypreservation *matching_algorithm_id matching_threshold* [*segmentation_result_in*|-] [*reference_segmentation_in*|-] [*col_out*|-]

Description

L'opérateur **passesstopologypreservation** permet de calculer 2 mesures pour évaluer la préservation de la topologie des régions d'un résultat de segmentation par comparaison avec une segmentation de référence (vérité terrain).

L'évaluation de la topologie en 2D se résume à l'analyse des trous internes aux régions. La préservation de la topologie est mesurée par deux erreurs :

- L'**erreur d'ajout de trou** qui rend compte de la proportion de faux trous détectés.
- L'**erreur de suppression de trou** qui rend compte de la proportion de trous non détectés.

Les valeurs d'erreurs sont dans l'intervalle [0..1], où 0 signifie aucune erreur et 1 le pire résultat. Une erreur d'ajout de trous de *x* signifie que *x * 100* pourcent des trous détectés sont des trous ajoutés. Une erreur de suppression de trous de *x* signifie que *x * 100* pourcent des trous n'ont pas été détectés. Le résultat est stocké dans la collection de sortie *col_out* qui contient les valeurs pour chacune des 2 mesures d'erreur.

Les mesures sont calculées sur la base d'un graphe de correspondance entre les segments du résultat de la segmentation et les régions de la segmentation de référence. Deux types de mise en correspondance sont possibles selon le paramètre *matching_algorithm_id* : le premier autorise la sur-segmentation et la sous-segmentation et le second ne permet que la mise en correspondance unique, un segment avec une région. Dans ce graphe, un segment *S* détecte une région *R* si la surface de recouvrement | R * S | est telle que :

```
| R * S |                                      | R * S |
--------- >= matching_threshold and  --------- >= matching_threshold
  | R |                                        | S |
```

Paramètres

- *matching_algorithm_id* : spécifie le numéro de l'algorithme de mise en correspondance à utiliser :
  - 0 : pour une correspondance de type 1-n et n-1. Un segment du résultat de la segmentation peut regrouper plusieurs régions de la référence (sous-segmentation), et une région de la référence peut être découpée en plusieurs segments du résultat de la segmentation

(sur-segmentation). Toutefois, un segment ou une région ne peuvent participer à la fois à une sur-segmentation et à une sous-segmentation.
    - 1 : pour une correspondance de type 1-1. Un segment du résultat de la segmentation ne peut être mis en correspondance qu'avec au plus une région de la référence, et une région de la référence ne peut être mise en correspondance qu'avec au plus un segment du résultat de la segmentation.
- *matching_threshold* : indique la proportion minimale de surface de recouvrement entre une région et un segment pour accepter une détection. C'est une valeur entre [0,1] où la valeur *x* correspond à un recouvrement minimum de (*x**100)%.

Entrées

- *segmentation_result_in* : une carte de régions contenant le résultat d'une segmentation.
- *reference_segmentation_in* : une carte de régions contenant la segmentation de référence.

Sorties

- *col_out* : une collection avec les 2 valeurs d'erreur.

Résultat

Retourne SUCCESS ou FAILURE en cas de problème.

Exemples

Évaluation de la préservation de la topologie des régions du résultat de la segmentation avec un taux de recouvrement minimum de 50% :

```
passesstopologypreservation 0 0.5 resultimages/algo001/tangram.pan groundtruths/expert001/tangram.pan errors.pan
pvisu errors.pan
```

Voir aussi

Evaluation, passessdetectionaccuracy, passessfragmentationconsistency, passessboundaryprecision, passessshapefidelity, passesssegmentationalgorithm, pranksegmentationalgorithms, pranksegmentationalgorithmsfromfolders

Prototype C++

```
Errc PAssessTopologyPreservation( const Reg2d
&segmentation_result_in, const Reg2d &reference_segmentation_in,
Collection &cold, const int matching_algorithm_id, const float
matching_threshold );
```

Auteur : Régis Clouard

pbarbremoval

Suppression des barbules sur leur longueur.

Synopsis

pbarbremoval *sens longueur* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pbarbremoval** consiste à supprimer les chaînes de contours **ouvertes** à partir de leur longueur. C'est le paramètre *sens* qui définit la relation sur la longueur entre inférieur, inférieur ou égal, égal, supérieur ou supérieur ou égal.

Une barbule est une chaîne continue en 8-connexité (ou 26 connexité en 3D) de pixels d'épaisseur 1 qui commence par un point terminal (1 voisin) et qui finit par une intersection avec une autre chaîne (> 1 voisins).

```
        /
 --------
        \
```

Attention: les points terminaux sont des points qui ne possèdent qu'un voisin. Il peut alors être utile de faire précéder cet opérateur d'opérateurs d'amincissement qui garantissent la 8-connexité (ex: ppostthinning).

Paramètres

- *relation* est une valeur entière de l'intervalle [-2,2] précisant la relation à la valeur de longueur.
  - *relation* = 2: barbules ≥ *longueur*.
  - *relation* = 1: barbules > *longueur*.
  - *relation* = 0: barbules = *longueur*.
  - *relation* = -1: barbules < *longueur*.
  - *relation* = -2: barbules ≤ *longueur*.
- La longueur *longueur* est comptée en nombre de pixels.

Entrées

- *im_in*: une image de type Uchar.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Le nombre de chaînes supprimées.

Exemples

Supprime les barbules des contours obtenus après une simple détection de contours:

```
psobel tangram.pan b.pan
pbinarization 45 1e30 b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
pbarbremoval 1 5 e.pan out.pan
pstatus
```

Voir aussi

Contour

Prototype C++

```
Errc PBarbRemoval( const Img2duc &im_in, Img2duc &im_out, int
relation, int length );
```

Auteur: Régis Clouard

pbellrescale

Retaille d'une image par l'algorithme de Bell.

Synopsis

pbellrescale *rescalex rescaley rescaleyz* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pbellrescale** utilise un noyau de convolution pour interpoler les valeurs des pixels de l'image d'entrée *im_in* afin de calculer les valeurs des pixels de l'image de sortie *im_out*.
L'interpolation consiste à pondérer l'influence des pixels d'entrée. Les poids sont dépendants de la position du pixel de sortie et sont donnés par l'algorithme de Bell:

```
       | 0.75-sqr(x) si -0.5<x<0.5
B(x) = | 0.5 * sqr(|x|-1.5) si -1.5< x <1.5
       | 0 sinon
```

Par exemple, si l'image est zoomée par 3, alors chaque pixel de sortie est donné par:

```
for i in [-2, 2]
  for j in [-2, 2]
      im_out[p.y][p.x] += B(i*scalex)*B(j*scaley)*im_in[p.y*scaley+j][p.x*scalex+i]
```

Pour zoomer une carte de régions ou un graphe, il faut utiliser l'operateur prescale.

Paramètres

- *rescalex*, *rescaley*, *rescalez* sont des réels positifs correspondant aux facteurs de retaille.
  Si les rescales sont > 1 alors il s'agit d'un agrandissement, s'ils sont < 1 alors il s'agit d'une réduction.
  rescalez est ignoré pour le cas des images 2D mais doit être donné.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Agrandissement de l'image d'un facteur 2 :

```
pbellrescale 2 2 0 tangram.pan a.pan
```

Réduction de l'image d'un facteur 2 :

```
pbellrescale 0.5 0.5 0 tangram.pan a.pan
```

Voir aussi

Transformation, plinearrescale, pbicubicrescale, planczosrescale, pmitchellrescale, prescale

Prototype C++

```
Errc PBellRescale( const Img2duc &im_in, Img2duc &im_out, float
rescaley, float rescalex );
```

Auteur: Régis Clouard

pbetagraph

Construction du béta-graphe d'un graphe.

Synopsis

pbetagraph *beta* [*-m mask*] [*gr_in*|-] [*gr_out*|-]

Description

Un béta graphe est un graphe dans lequel ont été supprimés tous les arcs considérés comme trop longs. Le principe de l'algorithme consiste à couper un arc si l'un de deux cercles d'intersection autour de ses deux sommets contient un autre sommet. Le cercle d'intersection entre un sommet i et un sommet j a pour centre et pour rayon:

```
centre=(1-beta/2)*p(i)+beta/2*p(j)
rayon=beta/2*distance(p(p(i),p(j))
```

On utilise ici la distance euclidienne entre les coordonnées des deux sommets.

beta donne la taille du rayon de l'intersection.

- Pour *beta*=1 on obtient alors le graphe de Gabriel.
- Pour *beta*=2 on obtient alors le graphe de voisins relatifs (GVR).

Les valeurs des sommets sont conservés dans *gr_out*. Par contre, les valeurs des poids sont mis à 1.

Paramètres

- Le paramètre *beta* est une valeur entière appartenant à l'intervalle [0..2] qui spécifie le rayon du cercle d'intersection.

Entrées

- *gr_in*: un graphe.

Sorties

- *gr_out*: un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule le graphe de Delaunay à partir des centres de gravité des objets dans l'image tangram.pan puis extrait le beta graphe.

```
pbinarization 90 1e30 tangram.pan a.pan
plabeling 8 a.pan r1.pan
pcenterofmass r1.pan r2.pan
pdelaunay r2.pan g2.pan
pbetagraph 1 g2.pan g3.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PBetaGraph( const Graph2d &gr_in, Graph &gr_out, float beta );
```

Auteur: François Angot

pbicubicrescale

Augmentation ou réduction de la taille d'une image par interpolation bicubique.

Synopsis

pbicubicrescale *rescalex rescaley rescaleyz* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pbicubicrescale** permet l'agrandissement ou la réduction de la taille d'une image par un facteur *rescalex* selon l'axe x, *rescaley* selon l'axe y et *rescalez* selon l'axe z (pour les images 3D). L'image est agrandie selon un axe si le facteur de rescale est > 1 et réduite si le facteur de rescale est >0 et <1.

Cette version utilise l'interpolation bicubique. Pour l'interpolation bicubique, la valeur du pixel de sortie est une moyenne pondérée des pixels dans le voisinage 4-4 autour du pixel.

Cet opérateur nécessite un temps d'exécution très long.

Pour retailler une carte de régions ou un graphe, il faut utiliser l'opérateur prescale.

Paramètres

- *rescalex*, *rescaley*, *rescalez* sont des réels positifs correspondant aux facteurs d'agrandissement. Si les rescales sont > 1 alors il s'agit d'un agrandissement, s'ils sont < 1 alors il s'agit d'une réduction.
  (*rescalez* est ignoré puisque l'opérateur ne fonctionne que pour les images 2D.)

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Agrandissement de l’image d’un facteur 2 :

```
pbicubicrescale 2 2 0 tangram.pan a.pan
```

Réduction de l’image d’un facteur 2 :

```
pbicubicrescale 0.5 0.5 0 tangram.pan a.pan
```

Voir aussi

Transformation, plinearrescale, prescale

Prototype C++

```
Errc PBicubicRescale( const Img2duc &im_in, Img2duc &im_out, float
rescaley, float rescalex );
```

Auteur: Régis Clouard

pbinarization

Seuillage binaire d'une image.

Synopsis

pbinarization *seuilb seuilh* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pbinarization** permet de sélectionner les pixels dont la valeur est comprise entre le *seuilb* et le *seuilh*. Dans l'image de sortie *im_out*, les pixels dont la valeur dans l'image d'entrée *im_in* est comprise dans l'intervalle [*seuilb*..*seuilh*] sont mis à 255, les autres sont mis à 0.

```
if im_in[p] ≥ low and im_in[p] ≤ high
then im_out[p]=255;
else im_out[p]=0;
```

Si *high* est inférieur à *low* alors **pbinarization** effectue la binarisation inverse :

```
if im_in[p] < high or im_in[p] > low
then im_out[p]=255;
else im_out[p]=0;
```

Pour les images couleurs ou multispectrales, le seuil est appliqué sur chaque bande. Pour les graphes, la binarisation est effectuée sur les valeurs de noeuds.

Paramètres

- *seuilb* et *seuilh* permettent de spécifier la zone de niveaux de gris à mettre en valeur, et l'intervalle des valeurs possibles est conditionnée par le type de l'image d'entrée. (ex: Img2duc [0..255], Img2dsl [-2147483648..+2147483648].)
  Si *seuilh* est supérieur à la valeur maximale du type des pixels alors c'est la valeur maximale qui est utilisée pour *seuilh* (ex: 255 pour Img2duc, +2147483648 pour Img2dsl).
  Si *seuilh* est inférieur à la borne inférieur alors c'est la valeur maxie qui est utilisée pour utilisée pour *seuilh*.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *im_out*: une image d'octets de même dimension que l'image d'entrée ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Segmente l'image tangram.pan en 2 classes que sont le fond et les pièces :

```
pbinarization 100 255 examples/tangram.pan out.pan
```

- Même résultat avec une borne supérieure supérieure à la valeur maximale:

```
pbinarization 100 1e30 examples/tangram.pan out.pan
```

- Même résultat avec une borne supérieure inférieure à la borne inférieure:

```
pbinarization 100 -1 examples/tangram.pan out.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PBinarization( const Img2duc &im_in, Img2duc &im_out, float
seuilb, float seuilh );
```

Auteur: Régis Clouard

pblend

Mélange d'images ou de graphes.

Synopsis

pblend *alpha* [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pblend** effectue un mélange de valeurs de pixel des deux images d'entrée *im_in1* et *im_in2*.

Si *im_in1* et *im_in2* sont des images alors la nouvelle image *im_out* est construite avec le mélange de chaque pxiel:

```
pixel(im_out) = alpha*pixel(im_in1) + (1-alpha)*pixel(im_in2);
```

Les deux images d'entrée doivent être du même type.

L'image de sortie *im_out* est du même type que les images d'entrée.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Si *im_in1* et *im_in2* sont des graphes alors le nouveau graphe *im_out* est construit avec le mélange des valeurs de noeud.

Paramaters

- *alpha* est un réel entre [0..1] qui représente le pourcentage d'image *im_in1* dans le mélange. Le pourcentage d'image *im_in2* est alors de 1-*aplha*.

Entrées

- *im_in1*: une image ou un graphe.
- *im_in2*: un objet du même type que *im_in1*.

Sorties

- *im_out*: un objet du même type que les entrées.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Effectue un moyennage entre les images a.pan et b.pan:

```
pblend 0.5 a.pan b.pan c.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PBlend( const Img2duc &im_in1, const Img2duc &im_in2, Img2dsf
&im_out, Float alpha );
```

Auteur: Régis Clouard

pblindedgeclosing

Fermeture de contours par poursuite de contours.

Synopsis

pblindedgeclosing *angle longueur* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pblindedgeclosing** consiste à fermer les contours donnés dans l'image *im_in*.

A partir des points terminaux, la poursuite se fait en recherchant un point contour devant. La recherche est limitées par le paramètre *angle* qui spécifie l'écart maximum autorisé avec la normale de la poursuite, et le paramètre *length* qui spécifie la longueur maximale autorisée.

Cet opérateur nécessite que les points terminaux n'aient qu'un seul voisin. Il peut donc être utile d'utiliser l'opérateur ppostthinning qui amincit les contours en ne gardant que la 8-connexité.

Paramètres

- Le paramètre *angle* permet de spécifier l'angle de recherche du point suivant. Il appartient à l'intervalle [0..2].
  - Si *angle*=0 alors la poursuite se fait dans la même direction que la fin du contour (0 degré de liberté).
  - *angle*=1 correspond à 0, 45 et -45 degrés.
  - *angle*=2 correspond à 0, 45, 90, -45, -90 degré.
- La *longueur* détermine la longueur maximale autorisée pour la poursuite.

Entrées

- *im_in*: une image 2D de type Uchar.
- *im_amp*: une image entière 2D.

Sorties

- *im_out*: une image 2D de Uchar.

Résultat

Retourne le nombre de contours fermés ou FAILURE.

Exemples

Ferme les contours obtenus par une simple detection de contours :

```
psobel tangram.pan b.pan
pbinarization 27 1e30 b.pan c.pan
pskeletonization 8 c.pan d.pan
ppostthinning d.pan e.pan
pblindedgeclosing 1 10 e.pan out.pan
pstatus
```

Voir aussi

Contour

Prototype C++

```
Errc PBlindEdgeClosing( const Img2duc &im_in, Img2duc &im_out, int angle, int longueur)
```

Auteur: Régis Clouard

pblockmatching

Estimation du mouvement entre deux images par mise en correspondance de blocs.

Synopsis

pblockmatching *block_size search_size ssd_min* [*-m mask*] [*im_in_ref*|-] [*im_in_dest*|-] [*im_out_dep*|-] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pblockmatching** permet de construire une image contenant l'estimation du mouvement entre deux images. L'image de sortie *im_out_dep* est une image multispectrale contenant le vecteur déplacement en chaque point: la première bande de l'image de sortie contient l'abscisse et la seconde bande l'ordonnée du vecteur déplacement.

La méthode d'estimation du mouvement par "block matching" consiste à établir une correspondance entre des blocs de pixels carrés de taille *block_size* d'une image de référence *im_in_ref* et des blocs de même taille d'une image de destination *im_in_dest*. On cherche alors le bloc de l'image de destination dans un voisinage *search_size* qui minimise la somme du carré des distances (SSD) :

```
SSD(u,v) = sum_{(x,y) in Block} [im_in_ref(x,y) - im_in_dest(x+u,y+v)]^2
```

Paramètres

- *block_size*: taille des blocs. Elle est souvent de 16.
- *search_size*: rayon de recherche des blocs similaires. Ce paramètre est à ajuster en fonction de la nature du mouvement entre les deux images.
- *ssd_min*: seuil de la SSD en dessous duquel on considère que le déplacement est non pertinent (le bloc n'est pas déplacé). Ce paramètre vise essentiellement à éviter les déplacements inutiles dans les régions homogènes.

Entrées

- *im_in_ref*: image 2D de référence.
- *im_in_dest*: image 2D de destination.

Sorties

- *im_out_dep*: image 2D multispectrale des déplacements (bande 0 : abscisse, bande 1 : ordonnée)

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcul du vecteur déplacement entre une image et son translaté :

```
ptranslation 0 17 tangram.pan tangram1.pan
pblockmatching 16 20 3 tangram.pan tangram1.pan deplacements.pan
pplotquiver 256 256 10 0.5 deplacements.pan out.pan
```

Voir aussi

Mouvement

Prototype C++

```
Errc pblockmatching(const Img2duc &im_in_ref, const Img2duc
&im_in_dest, Imx2dsf &im_out_dep, short block_size_x, short
search_size, short ssd_min);
```

Auteurs: G. Née - Y. Pitrey Helpiquet - S. Jéhan Besson

pbmp2pan

Conversion d'une image BMP en image Pandore.

Synopsis

pbmp2pan *im_in* [*im_out*|-]

Description

L'opérateur **pbmp2pan** permet de convertir une image de type *bmp* en un fichier *Pandore*.
Le fichier résultant est de type :

- Img2duc si le nombre de bit pour coder un pixel est égal à 1, 4 ou 8.
- Imc2duc si le nombre de bit est égal à 24.

Entrées

- *im_in*: une image BMP.

Sorties

- *im_out*: une image Pandore.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image BMP image.bmp en image Pandore :

```
pbmp2pan image.bmp image.pan
```

Voir aussi

Conversion, ppan2bmp

Prototype C++

```
Errc PBmp2Pan( const FILE* fdin, Pobject** objout );
```

Auteur: Régis Clouard

pborsotti

Calcul du critère de qualité basé sur le nombre, l'aire et la variance des régions.

Synopsis

pborsotti [*-m mask*] [*rg_in*|-] [*im_in*|-]

Description

L'opérateur **pborsotti** calcule un critère de qualité pour l'évaluation d'une segmentation d'une image de niveaux de gris, de couleur ou multispectrale tel que défini par M. Borsotti*.

La mesure est définie à partir de trois critères :

- les regions doivent être uniformes et homogènes ;
- l'intérieur des regions doit être simple sans trop de petits trous ;
- les régions adjacentes doivent présentées des valeurs différentes pour des caractéristiques d'uniformité.

La mesure est calculée comme suit :

$$F(I) = (1/(1000*A)) * \sqrt{N} * \sum_R [(ei^2 / (1+\log(Ai)) + (R(Ai)/ Ai)^2)]$$

where

- A est la surface totale des régions.
- Ai est la surface de la région i.
- N est le nombre de régions.
- R(Ai) est le nombre de régions qui ont la même surface que Ai.
- ei est défini comme la somme des distances euclidiennes entre le vecteur couleur du pixel de la région i et le vecteur couleur attribuée à la région i.

L'équation précédente est composée de trois termes :

1. un facteur de normalization qui prend en compte la taille de l'image ;
2. un facteur de pénalization pour un sous-segmentation ;
3. la somme est composée de deux partie :
   - une pénalisation pour les petites régions et les régions hétérogènes ;
   - une pénalisation pour les régions de petite taille (sur-segmentation).

Plus la valeur est petite, meilleure est la segmentation.

Attention: Les régions de label=0 ne sont pas prises en compte pour la mesure.

Entrées

- *rg_in*: une carte de région.
- *im_in*: une image.

Résultat

Retourne un réel positif.
(Utiliser `pstatus` pour récupérer cette valeur).

Exemples

Calcule la valeur du critère de Borsotti pour une simple segmentation par binarisation:

```
pbinarization 80 1e30 tangram.pan i1.pan
plabeling 8 i1.pan i2.pan
pborsotti i2.pan tangram.pan
pstatus
```

Voir aussi

Evaluation

Prototype C++

```
Errc PBorsotti( const Reg2d &rg_in, const Imc2duc &im_in );
```

Reference

* M. Borsotti, P. Campadelli, R. Schettini, "Quantitative evaluation of color image segmentation results", *Pattern Recoginition Letters*, 19:741-747, 1998.

Auteur: Régis Clouard

pboundary

Localisation des points de frontière des régions.

Synopsis

pboundary *connexite* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pboundary** construit une image binaire formée par les points de frontière entre des régions de valeurs de pixel ou de label différentes.

Un point appartient à une frontière lorsqu'au moins un des voisins n'a pas la même valeur de label ou de pixel que lui. Dans le cas d'une carte de régions, la région de label=0 n'est pas considérée comme une région et n'a donc pas de frontière propre.

Une frontière est forcément un contour fermé.

La frontière entre 2 régions non nulles ne pouvant être mise entre 2 pixels, celle-ci est doublée sur chacune des régions. Ceci fait que la frontière entre 2 régions non nulles a une taille de 2 pixels.

Les frontières de l'image de sortie *im_out* sont marquées par des pixels de valeur 255 reposant sur un fond de valeur égal à 0. L'image de sortie est forcément de type Uchar (Img2duc ou Img3duc).

Pour les graphes, les sommets ayant 0 ou 1 voisin sont considérés comme des points de bordures et donc deviennent des points de frontières dans l'image de sortie.

Paramètres

- *connexite* définit la notion de voisinage: 4, 8 pour le 2D, ou 6 ou 26 pour le 3D.
  Ce paramètre est ignoré pour les graphes.

Entrées

- *im_in*: une image de niveaux de gris ou une carte de régions.

Sorties

- *im_out*: une image binaire.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Localise les frontières des régions de la carte a.pan (ajoute une frontière sur le bord) :

```
pboundary 8 a.pan b.pan
psetborder 1 1 1 1 0 0 255 b.pan c.pan
```

Voir aussi

Région

Prototype C++

```
Errc PBoundary( const Reg2d &im_in, Img2duc &im_out, int connexite
);
```

Auteur: Régis Clouard

pboundarylabeling

Etiquetage en régions d'une image de contours fermés.

Synopsis

pboundarylabeling [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pboundarylabeling** consiste à marquer par une même région un ensemble de pixels connexes de *im_in* délimitée par un contour fermé.

Un contour fermé est une séquence en 8-connexité de pixels non nuls qui boucle sur elle-même. Tous les pixels à l'intérieur (nuls ou non nuls) sont englobés dans la région sauf s'ils forment un autre contour fermé.

Les contours sont incorporés dans la région la plus à gauche.

Une région de la carte de région *rg_out* est définie par une ensemble de pixels ayant exactement le même label. Chaque région reçoit un numéro de label unique et minimal mais non nul.

Entrées

- *im_in*: une image 2D.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de régions construites.

Exemples

Fusionne les régions obtenues après une division par quadtree :

```
puniformityquadtree 0.9 tangram.pan a.pan
prg2gr a.pan b.pan
pboundarymerging -1 4 a.pan b.pan tangram.pan c.pan d.pan
```

Voir aussi

Segmentation, plabeling

Prototype C++

```
Errc PBoundaryLabeling( const Img2duc &im_in, Reg2d &rg_out );
```

Auteur: Régis Clouard

pboundarymerging

Fusion prioritaire de régions selon le contraste aux frontières.

Synopsis

pboundarymerging *nb_fusion seuil* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

L'opérateur **pboundarymerging** permet de fusionner les régions de la carte de régions *rg_in* selon la valeur de contraste aux frontières.

La notion de voisinage entre les régions est détenue par le graphe *gr_in*.

Le principe de l'algorithme est le suivant:
Pour chaque région de la carte de régions, on calcule le contraste à la frontière de ses voisines. Si la valeur de contraste est inférieure au *seuil* donnée. paramètre, alors les régions de part et d'autre de la frontière sont fusionnées.
On utilise ici l'algorithme de croissance prioritaire qui consiste à fusionner à chaque fois les 2 régions dont la différence est la plus faible.

im_in peut être une image de gradient ou une image d'intensité.

Le contraste est calculé par:

```
contraste(R1,R2)= 1/N * sum(max(C(s,t), t in V(s) et t in R2 et s in R1))
avec C(s,t)= | im_in[s] - im_in[t] |
où N = nombre de pixels de la frontière.
```

Paramètres

- *nb_fusion* permet de spécifier le nombre de fusion à effectuer (la valeur -1 signifie d'ignorer ce paramètre et d'exécuter l'algorithme tant qu'il y a des fusions possibles).
- *seuil* permet de spécifier la tolérance maximale sur l'écart de contraste sur la frontière. Les valeurs appartiennent à l'intervalle [0..nombre de niveaux de gris].

Entrées

- *rg_in*: une carte de régions.
- *gr_in*: un graphe.
- *im_in*: une image.

Sorties

- *rg_out*: une carte de régions.
- *gr_out*: un graphe.

Résultat

Retourne le nombre de fusions effectuées.

Exemples

Fusionne les régions issues d'une partition de l'image tangram.pan :

```
puniformityquadtree 0.9 tangram.pan a.pan
prg2gr a.pan b.pan
pboundarymerging -1 4 a.pan b.pan tangram.pan c.pan d.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PBoundaryMerging( const Reg2d &rg_in, const Graph2d &gr_in,
const Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, long
nb_fusion, Uchar seuil );
```

Auteur: Laurent Quesnel

pboundaryregularization

Régularisation des frontières des régions.

Synopsis

pboundaryregularization *halfsize* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pboundaryregularization** reconstruit la forme des régions de la carte d'entrée *rg_in* en régularisant plus ou moins leur frontière, c'est-à-dire en rendant les frontières plus lisses. Le paramètre *halfsize* permet de régler la force de la régularisation.

Paramètres

- *halfsize* donne la demi-taille de la fenêtre d'analyse de la regularité. Il correspond approximativement à l'écart en pixels toléré pour les variation des contours. Plus ce paramètre est grand plus la régularisation sera importante.

Entrées

- *rg_in*: une carte de régions (2D ou 3D).

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pboundaryregularization 3 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PBoundaryRegularization( const Reg2d &, Reg2d &im_out, int
halfsize );
```

Auteur: Régis Clouard

pboundingbox

Calcul du rectangle exinscrit des régions.

Synopsis

pboundingbox [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pboundingbox** construit une carte de régions avec les rectangles exinscrits des régions de la carte de régions d'entrée *rg_in*.

A partir de la carte de régions *rg_in*, les coordonnées extrémales de chaque région sont calculées et le rectangle ainsi défini est rempli avec la valeur de label de la région. La carte *rg_out* est alors composée de parallelépipèdes rectangles (pleins) correspondant aux dimensions extrêmes des régions d'entrée.

Les labels des rectangles de *rg_out* reprennent ceux des régions correspondantes dans *rg_in*.

Attention: Il peut y avoir superposition des régions. Dans ce cas c'est la région de plus fort label qui sera conservée.

Entrées

- *rg_in*: une carte de régions.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit le rectangle exinscrit autour des régions de la carte rin.pan

```
pboundingbox rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PBoundingBox( cont Reg2d &rg_in, Reg2d &rg_out );
```

Auteur: François Angot

pbutterworthfilter

Génère un filtre passe-bas, passe-haut, coupe-bande ou passe-bande de Butterworth.

Synopsis

pbutterworthfilter [-m mask] *ncol nrow ndep highpass cutin cutoff order* [*im_out*|-]

Description

L'opérateur **pbutterworthfilter** génère un filtre passe-bas, passe-haut, coupe-bande ou passe-bande de Butterworth.

Si *ndep<1* le filtre *im_out* est une image 2D de Float avec size nrow*ncol sinon le filtre *im_out* est une image 3D de Float avec la taille ndep*nrow*ncol.

Le filtre passe-bas de Butterworth coupe les hautes fréquences des composantes de la transformée de Fourier qui sont à une distance supérieure à la distance spécifiée D0 (la valeur *cutoff*) à partir de l'origine du centre de la transformation.

Le type du filter est donné par les deux paramètres *highpass* et *cutin*:

- highpass=0 et cutin=0 : filtre passe-bas
- highpass=1 et cutin=0 : filtre passe-haut
- highpass=0 et cutin=1 : filtre coupe-bande
- highpass=1 et cutin=1 : filtre passe-bande.

La fonction de transfert d'un filtre 2D passe-bas de Butterworth d'ordre n avec une fréquence de coupe à la distance D0 de l'origine est définie par :

$$Hlp(u,v) = \frac{1}{1 + [\, D(u,v)/D0]^{2n}}$$

où D(u,v) est la distance du point (u,v) à l'origine :

$$D(u,v)=\mathrm{sqrt}((u-M/2)^2 + (v-N/2)^2)$$

où N est le nombre de lignes et M est le nombre de colonnes.

La fonction de transfert d'un filtre 2D passe-haut de Butterworth est définie par :

$$H(u,v) = 1 - Hlp(u,v)$$

La fonction de tranfert d'un filtre coupe-bande est définie par:

$$H(u,v) = \frac{1}{1 + \left[\frac{D(u,v)W}{D^2(u,v) - D0^2}\right]^{2n}}$$

où W est l'épaisseur des bandes = cutof-cutin et D0 est la rayon=(cutin+cutoff)/2.

Paramètres

- *ncol*, *nrow*, *ndep* spécifie la taille de l'image de sortie. Si *ndep*<1 alors la sortie est une image 2D sinon une image 3D.
- *highpass* est utilisée. conjonction avec le paramètre *cutin*. Il spécifie le type de filtre :
  - highpass=0 et cutin=0 : filtre passe-bas
  - highpass=1 et cutin=0 : filtre passe-haut
  - highpass=0 et cutin=1 : filter coupe-bande
  - highpass=1 et cutin=1 : filtre passe-bande
- *cutin* est la fréquence de coupe du filtre D0 en cas de filtre coupe-bande ou bande-passe. Dans ce cas, l'épaisseur des bandes = cutoff-cutin et D0=(cutoff+cutin)/2.
- *cutoff* est la fréquence de coupe du filtre D0. C'est un réel positif dans l'intervalle]0.. sqrt(M*m+N*n)/2]. Il correspond à la distance euclidienne de la bande au centre de l'image. Plus cutoff est élevé, plus le filtrage passe-bas est faible et plus le filtrage passe_haut est fort.
- *order* est l'ordre du filtre. Plus l'ordre est élevé plus la transition est franche. C'est un entier >= 1. Une valeur typique est 1.

Sorties

- *im_out*: une image Float (Img2dsf ou Img3dsf).

Résultat

Retourne SUCCESS ou FAILURE en cas de mauvais paramétrage.

Exemples

Effectue un filtrage passe-bas de Butterworth :

```
psetcst 0 tangram.pan i1.pan
pfft tangram.pan i1.pan i2.pan i3.pan
pbutterworthfilter 256 256 0 0 0 50 2 i4.pan
pmult i2.pan i4.pan i5.pan
pmult i3.pan i4.pan i6.pan
pifft i5.pan i6.pan out.pan i8.pan
```

Effectue un filtrage passe-haut de Butterworth :

```
psetcst 0 tangram.pan i1.pan
pfft tangram.pan i1.pan i2.pan i3.pan
pbutterworthfilter 256 256 0 1 0 50 2 i4.pan
pmult i2.pan i4.pan i5.pan
pmult i3.pan i4.pan i6.pan
pifft i5.pan i6.pan out.pan i8.pan
```

Effectue un filtrage coupe-bande de Butterworth :

```
psetcst 0 tangram.pan i1.pan
pfft tangram.pan i1.pan i2.pan i3.pan
pbutterworthfilter 256 256 0 0 25 50 2 i4.pan
pmult i2.pan i4.pan i5.pan
pmult i3.pan i4.pan i6.pan
pifft i5.pan i6.pan out.pan i8.pan
```

Effectue un filtrage passe-bande de Butterworth :

```
psetcst 0 tangram.pan i1.pan
pfft tangram.pan i1.pan i2.pan i3.pan
pbutterworthfilter 256 256 0 1 25 50 2 i4.pan
pmult i2.pan i4.pan i5.pan
pmult i3.pan i4.pan i6.pan
pifft i5.pan i6.pan out.pan i8.pan
```

Voir aussi

Domaine Fréquentiel, pifft pfftshift

Prototype C++

```
Errc PButterworthFilter( Img2dsf &im_out, int ndep, int nrow, int
ncol, int highpass, float cutin, float cutoff, int order);
```

Auteur: Régis Clouard

pcenterofmass

Localisation des centres de gravité de régions.

Synopsis

pcenterofmass [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pcenterofmass** permet de localiser les centres de gravité des régions dans la carte *rg_in* par un point dans la carte de régions de sortie *rg_out*. Les centres de gravité de *rg_out* conservent les mêmes labels que les régions d'origine.

Le centre de gravité d'une région de taille N pixels est calculé par:

```
gx = SOMME(x[Ri])/N
gy = SOMME(y[Ri])/N
```

Attention: les centres de gravité ne se trouvent pas nécessairement sur la région. De ce fait, si 2 centres de gravité sont superposés, il ne restera que le dernier sur la carte de région finale.

Entrées

- *rg_in*: une carte de régions.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Localise les centre de gravité des régions de la carte rin.pan :

```
pcenterofmass rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PCenterOfMass( const Reg2d &rg_in, Reg2d &rg_out );
```

Auteur: Régis Clouard

pchanda

Multiseuillage de l'image par analyse de la matrice de co-occurences selon Chanda, Chauduri et Majumder.

Synopsis

pchanda *length* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pchanda** permet de construire une image seuillée à partir de l'algorithme de Chanda, Chauduri et Majumder basé sur l'analyse de matrice de co-occurrence.

Le calcul pour chaque niveau de gris n de [0..N-1] de la mesure de contraste moyen est fait par :

```
        Somme(Somme((Tkl)*(Tkl))   Somme(Somme((Tpq * Tpq)))
C(n) =------------------------- + ----------------------
        Somme(Somme(Tkl))          Somme(Somme(Tpq))
```

avec Tkl la matrice de co-occurence Tkl non symétrique définie avec le voisin 6.
avec k=[0..n], l=[n+1..N-1]
avec p=[n+1..N-1], q=[0..n]
La recherche des maxima locaux de Constraste(n) est faite sur toute la plage de niveaux de gris de part et d'autre du niveau de gris n.

Remarque: Cet opérateur ne fonctionne que sur des images de Char parce qu'il faut que les transitions Tkl soient significatives (ie, nombre de (k,l) restreints). Il faut donc s'arranger pour transformer les autres types d'images en image de Uchar.

L'image de sortie *im_out* est contruite avec les seuils détectés, telle que :

`im_out[y][x]=seuil[k] si seuil[k-1]<im_out[y][x]<=seuil[k].`

Le dernier seuil est égal à la valeur maximale 255.

Paramètres

- *length* définit la plage de recherche des minima de la fonction énergie. Il est défini en unité de nievaux de gris. Plus il est grand, moins il y a de seuils.

Entrées

- *im_in*: une image 2D d'octets (Img2duc, Img3duc).

Sorties

- *im_out*: une image 2D d'octets (Img2duc, Img3duc).

Résultat

Retourne le nombre de classes détectés.

Exemples

Segmente l'image tangram.pan et affiche le nombre de classes detectées :

```
pchanda 20 tangram.pan out.pan
pstatus
```

Voir aussi

Seuillage

Prototype C++

```
Errc PChanda( const Img2duc &im_in, Img2duc &im_out, int length );
```

Auteur: Régis Clouard

pcliparea

Sélection d'une zone d'image, de carte de région ou de graphe.

Synopsis

pcliparea *x y z width height depth* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pcliparea** permet de sélectionner une région rectangulaire de l'entrée *im_in*.

Les valeurs hors du rectangle sont mises à 0 dans la sortie *im_out* et les autres sont recopiées.

Paramètres

- *x*, *y*, *z* spécifient les coordonnées du rectangle et *width*, *height*, *depth* la taille du rectangle. Si *width* (respectivement *height* ou *depth*) est < 1 ou à une valeur qui dépasse la largeur de l'image alors la valeur est considérée comme la valeur maximale.
  Pour les objets 2D, *z* et *depth* doivent être donnés mais sont ignorés.

Entrées

- *im_in*: une image, une carte de régions ou un graphe.

Sorties

- *im_out*: un objet de même type que l'objet d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Sélection de la zone (10,10,246,50) de l'image tangram.pan sachant que l'image est de taile 256x256 :

```
pcliparea 10 10 -1 50 0 0 tangram.pan a.pan
```

Voir aussi

Utilitaire

Prototype C++

```
Errc PClipArea( const Imx3d &ims, Imx3d &imd, const int z, const int
y, const int x, int width, int height, int depth);
```

Auteur: Régis Clouard

pclipvalues

Ecrêtage des valeurs de pixels

Synopsis

pclipvalues *low high* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pclipvalues** écrête les valeurs de pixel de l'image d'entrée *im_in* entre les valeurs spécifiées par les bornes [*low*..*high*]. Les pixels avec une valeur inférieure à *low* sont remplacées par *low*; les pixels avec une valeur supérieure à *high* sont remplacées par *high*.

Plus formellement, *im_out* est construite en utilisant l'algorithme suivant sur chaque pixel p:

```
if (im_in[p] > high) im_out[p]=high
else if (im_in[p] < low) im_out[p]=low
     else im_out[p]=im_in[p];
```

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Paramètres

- *low* et *high* spécifie le domaine de valeur de l'image de sortie. Les valeurs sont relatives au type de l'image d'entrée (par exemple Img2duc [0..255],Img2dsl [-2147483648..+2147483648]).

  Note: si *min* > *max* alors min et max sont respectivement affecté avec le minimum et le maximum des valeurs du type (par exemple, 0 et 255 pour les images de Uchar ou +2147483648 pour Img2dsl).

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de même type que l'image d'entrée *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Rehaussement de contraste de l’image tangram.pan en utilisant la technique du "unsharp masking". L’image réhaussée est construite en ajoutant l’image filtrée par un filtre passe-haut. L’image filtrée est construite en soustrayant l’image initiale avec une version lissée de celle-ci. A la fin, seuls les pixels entre 0 et 255 sont conservés.

```
pim2sf tangram.pan i1.pan
pgauss 0.8 i1.pan i2.pan
psub i1.pan i2.pan i3.pan
pmultcst 0.7 i3.pan i4.pan
padd i1.pan i4.pan i5.pan
pclipvalues 0 255 i5.pan mean.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PClipValues( const Img2duc &im_in, Img2duc &im_out, float low,
float high );
```

Auteur: Régis Clouard

pclosedcontourselection

Sélection des chaînes de contours fermées sur leur longueur.

Synopsis

pclosedcontourselection *relation longueur* [*-m mask*] [*im_in*|*-*] [*im_out*|*-*]

Description

L'opérateur **pclosedcontourselection** consiste à sélectionner dans une image de contours toutes les **chaînes fermées** ayant une longueur:

- supérieure : si *relation > 0*
- inférieure : si *relation < 0*
- égale : si *relation = 0*

à la valeur de *longueur* donnée en paramètre et comptée en nombre de pixels.

Une boucle (ou une chaîne fermée) est une séquence continue de pixels non nuls, d'épaisseur 1 pixel, et qui boucle sur elle-même.

Attention: Si les contours ne sont pas tous d'épaisseur 1, l'opérateur peut avoir un comportement imprévisible. Attention aussi aux problèmes de coins qui peuvent engendrer de fausses détections de contours fermés.
Il faut généralement faire précéder cet opérateur d'opérateurs d'amincissement qui garantissent la 8-connexité (ex: ppostthinning).

Paramètres

- Le *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de longueur.
  - *relation* = 3: contours fermés avec la longueur maximale.
  - *relation* = 2: contours fermés >= *longueur*.
  - *relation* = 1: contours fermés > *longueur*.
  - *relation* = 0: contours fermés = *longueur*.
  - *relation* = -1: contours fermés < *longueur*.
  - *relation* = -2: contours fermés <= *longueur*.
  - *relation* = -3: contours fermés avec la longueur minimale.
- La longueur *longueur* est comptée en nombre de pixels.

Entrées

- *im_in*: une image de type Uchar.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Le nombre de chaînes sélectionnées.

Exemples

Sélectionne les plus longs contours fermés dans un ensemble de contours obtenus par une simple détection d econtours :

```
psobel tangram.pan b.pan
pbinarization 45 1e30 b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
pclosedcontourselection 3 5 e.pan out.pan
pstatus
```

Voir aussi

Contour

Prototype C++

```
Errc PClosedContourSelection( const Img2duc &im_in, Img2duc &im_out,
int relation, int longueur );
```

Auteur: Régis Clouard

pcmyk2rgb

Changement d'espace couleur de l'espace CMYK (Cyan-Magenta-Yellow-Key) à l'espace RVB.

Synopsis

pcmyk2rgb [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pcmyk2rgb** permet de changer d'espace couleur de l'espace Cyan, Magenta, Yellow, Key, vers l'espace Rouge-Vert-Bleu

La conversion procède selon l'algorithme suivant:
The red (R) color is calculated from the cyan (C) and black (K) colors:

```
R = 255 * (1-C/255) * (1-K/255)
```

The green color (G) is calculated from the magenta (M) and black (K) colors:

```
G = 255 * (1-M/255) * (1-K/255)
```

The blue color (B) is calculated from the yellow (Y) and black (K) colors:

```
B = 255 * (1-Y/255) * (1-K/255)
```

Entrées

- *im_in*: une image multispectrale CMYK.

Sorties

- *im_out*: une image couleur.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit butterfly.pan de rgb en cmyk, puis de cmyk en rgb:

```
prgb2cmyk butterfly.pan a.pan
pcmyk2rgb a.pan b.pan
```

Voir aussi

Color, prgb2cmyk

Prototype C++

```
Errc PCmyk2Rgb( const Imx2duc &im_in, Imx2duc &im_out );
```

Auteur: Régis Clouard

pcol2csv

Conversion d'une collection en un fichier texte au format csv.

Synopsis

pcol2csv [*col_in*|-] [*file_out*]

Description

L'opérateur **pcol2csv** permet de convertir le contenu d'une collection contenant des valeurs et des tableaux en un fichier texte. Les fichiers de type CSV (Comma Separated Values) sont des fichiers texte qui contiennent une liste tabulée de données en ASCII qui peut être utilisée dans des logiciels de type tableurs.

La sortie suit le format suivant:

```
label1; label2; label3;
val11;  val21;  val31;
  ..      ..     ..
val1n;  val2n;  val3n;
```

où le séparateur est le point virgule.

Le nom du fichier texte *file_out* est optionnel. S'il est omis, le contenu de la collection s'affiche sur la sortie standard.

Entrées

- *col_in*: les fichiers collection.

Sorties

- *file_out*: un fichier texte.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche le contenu de la collection col.pan :

```
pobject2col foo tangram.pan col.pan
pcolsetvalue foo Float 10.5 col.pan col.pan
pfile col.pan
pcol2csv col.pan data.csv
```

Voir aussi

Collection

Prototype C++

```
Errc PCol2Csv( const Collection &col_in_out, FILE *fd );
```

Auteur: Alexandre Duret-Lutz

pcol2txt

Conversion d'une collection en un fichier texte.

Synopsis

pcol2txt [*col_in*|-] [*file_out*]

Description

L'opérateur **pcol2txt** permet de convertir le contenu d'une collection contenant des valeurs et des tableaux en un fichier texte. Le nom du fichier texte *file_out* est optionnel. S'il est omis, le contenu de la collection s'affiche sur la sortie standard.

Remarque: les valeurs de type Char ou Uchar sont considérées comme des entiers courts. Ceci fait que les chaînes de caractères (de type Array:Char) sont affichées dans le fichier texte sous la forme de leur code ASCII (A=65, B=66, ...).

Entrées

- *col_in*: les fichiers collection.

Sorties

- *file_out*: un fichier texte.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche le contenu de la collection col.pan :

```
pobject2col foo tangram.pan col.pan
pcolsetvalue foo Float 10.5 col.pan col.pan
pfile col.pan
pcol2txt col.pan
```

Voir aussi

Collection

Prototype C++

```
Errc PCol2Txt( const Collection &col_in_out, FILE *fd );
```

Auteur: Alexandre Duret-Lutz

pcolcatenateitem

Concaténation des attributs de deux collections.

Synopsis

pcolcatenateitem [*col_in1*|-] [*col_in2*|-] [*col_out*|-]

Description

L'opérateur **pcolcatenateitem** permet de créer la collection *col_out* à partir des attributs des deux collections sources. Si un attribut porte le même nom dans les deux collections et que les types sont compatibles, alors un tableau contenant la concaténation des valeurs est créé, sinon l'attribut de *col_in1* est préféré.

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Concatène les 2 collections c1.pan et c2.pan dans la collection col.pan puis extrait les images dans les foo.pan et bar.pan:

```
pobject2col foo tangram.pan c1.pan
pobject2col bar parrot.pan c2.pan
pcolcatenateitem c1.pan c2.pan col.pan
pcolgetimages col.pan
```

Voir aussi

Collection

Prototype C++

```
Errc PColCatenateItem( const Collection &col_in_out1, const
Collection &col_in2 );
```

Auteur: Alexandre Duret-Lutz

pcolgetimages

Extraction des images d'une collection.

Synopsis

pcol2images [*col_in*|-]

Description

L'oéperateur **pcolgetimages** extrait toutes les images de la collection d'entrée *col_in*. Chaque image est sauvée dans le fichier nommé à partir du nom dans la collection suffixée ".pan".

Par exemple, si la commande "pfile col.pan" retourne :

```
Creator   : pandore
Date      : 2005/04/13
Type      : Collection (Bundle of elements)
Number of elements : 3
        attr       (8 elements)   Array:Ulong
        image2d    (1 element)    Pobject:Img2duc
        image3d    (1 element)    Pobject:Img3duc
```

alors la commande "pcolgetimages col.pan" construit :

```
image2d.pan
image3d.pan
```

L'opérateur pcolgetobject peut être utilisé pour extraire une seule image d'une collection.

Entrées

- *col_in*: une collection.

Résultat

Retourne SUCCESS ou FAILURE si au moins une image ne peut pas être sauvée.

Exemples

Ajoute les 2 images tangram.pan et femme.pan dans une collection puis extrait ces images dans deux autres fichiers :

```
pobject2col image1 tangram.pan c1.pan
pobject2col image1 femme.pan c2.pan
concateneattribut c1.pan c2.pan col.pan
pcol2images col.pan
```

Voir aussi

Collection

Prototype C++

```
Errc PColGetImages( Collection &col_in );
```

Auteur: Nicolas Briand

pcolgetobject

Extraction d'un objet Pandore d'une collection.

Synopsis

pcolgetobject *name* [*col_in*|-] [*obj_out*|-]

Description

L'opérateur **pcolgetobject** extrait l'objet Pandore nommé *name* dans la collection d'entrée *col_in* pour créer l'objet de sortie *obj_out*. L'objet Pandore peut être une image, une carte de régions un graphe ou même une collection.

Paramètres

- *name* est le nom de l'objet dans la collection.

Entrées

- *col_in*: une collection.

Sorties

- *obj_out*: un fichier Pandore.

Résultat

Retourne SUCCESS ou FAILURE si l'objet n'existe pas ou n'est pas un objet Pandore.

Exemples

Extrait l'image nommée foo dans la collection col.pan dans le fichier a.pan :

```
pobject2col foo tangram.pan col.pan
pcolgetobject foo col.pan a.pan
pfile a.pan
```

Voir aussi

Collection, pcolgetimages

Prototype C++

```
Errc PColGetObject( Collection &col_in, Pobject * &obj_out, const
std::string &name );
```

Auteur: Alexandre Duret-Lutz

pcolgetvalue

Extraction de la valeur d'un numérique dans une collection.

Synopsis

pcolgetvalue *name* [*col_in*|-]

Description

L'opérateur **pcolgetvalue** permet de récupérer dans la valeur de sortie la valeur de l'attribut numérique nommé *nom_attribut* dans la collection *col_in*.

La valeur de sortie est ensuite récupérable par l'opérateur **pstatus**.

Paramètres

- *name* est un identificateur alphabétique sans accent, sans espace et sans ponctuation.

Entrées

- *col_in*: une collection.

Résultat

Retourne la valeur numérique de l'attribut ou FAILURE.

Exemples

Ajoute la valeur réelle 10.5 dans la collection col.pan avec le nom "foo" puis vérifie si la valeur est dans la collection :

```
pobject2col image tangram.pan col.pan
pcolsetvalue foo Float 10.5 col.pan col.pan
pfile col.pan
pcolgetvalue foo col.pan
pstatus
```

Voir aussi

Collection

Prototype C++

```
Errc PColGetValue( const std::String name, Collection &col_in );
```

Auteur: Régis Clouard

pcolorcube

Visualisation de la répartition des couleurs d'une image dans un cube (représentant l'espace couleur).

Synopsis

cubecouleurs *x y z* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pcolorcube** permet de créer une image synthétique couleur pour visualiser sous la forme d'un cube RVB la répartition des couleurs de l'image couleur *im_in*.

Le cube est vu en 2D selon la position d'un observateur données par les coordonnées *x*,*y*,*z* par rapport à l'origine du cube.

L'axe principal de répartition des données apparaît sous la forme d'une droite de couleur blanche.

Paramètres

- *x*,*y* et *z* spécifient la position de l'observateur par rapport à l'origine du repère du cube.

Entrées

- *im_in*: une image couleur.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche le cube couleur de l'image parrot.pan:

```
pcolorcube 0 0 0 parrot.pan a.pan
```

Voir aussi

Visualisation

Prototype C++

```
Errc PCubeCouleurs( const Imc2duc &im_in, Imc2duc &im_out, int x,
int y, int z );
```

Auteur: Olivier Lezoray

pcolorize

Colorisation de régions à partir de sa valeur moyenne.

Synopsis

pcolorize [*-m mask*] [*im_in*|-] [*rg_in*|-] [*im_out*|-]

Description

L'opérateur **pcolorize** permet de créer une image *im_out* à partir de l'image *im_in* et de la carte de régions associée *rg_in*. La couleur de chaque pixel de l'image de sortie *im_out* est la couleur moyenne de la région à laquelle il appartient.

L'image de sortie *im_out* est du même type que l'image d'entrée *im_in*.

Entrées

- *im_in*: une image.
- *rg_in*: une carte de régions.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Colorise les régions de la carte de région reg.pan avec leur moyenne intérieure :

```
pcolorize in.pan reg.pan out.pan
```

Voir aussi

Visualisation

Prototype C++

```
Errc PColorize( const Imc2duc &im_in, const Reg2d &rg_in, Imc2duc
&im_out );
```

Auteur: Olivier Lezoray

pcolorquantization

Réduction du nombre de couleurs utilisées pour coder une image

Synopsis

pcolorquantization *color_number color_space* [-m mask] [*im_in*|-] [*im_out*|-]

Description

La quantification couleur a pour but de réduire le nombre de couleurs en minimisant la différence de sensation visuelle. L'image résultat *im_out* est une image de couleur avec le nombre de couleurs spécifié ou moins.

L'algorithme est basé sur la minimisation de l'erreur quandratique globale après réaffectation des couleurs :

```
Erreur = sum_{ij} ||c(i,j) - c'(i,j)||^2
```

o&uagrave; c(i,j) est la couleur du pixel (i,j) dans l'image originale et c'(i,j) est la couleur dans l'image transformée. La couleur d'un pixel dépend de l'espace considéré.

Cet opérateur est souvent utilisé comme prétraitement avant une segmentation.

Paramètres

- *color_number* : définit le nombre de couleurs désirées en sortie. C'est un entier > 0.
- *color_space* : une valeur entre [0,6] qui définit l'espace de segmentation le plus approprié :
  0. Espace R, G, B.
  1. Espace I1, I2, I3.
  2. Espace H1, H2, H3.
  3. Espace L, u, v.
  4. Espace Y, I, Q.
  5. Espace des vecteurs propres.
  6. Espace K1, K2, K3.

Entrées

- *im_in*: une image couleur.

Sorties

- *im_out*: une image couleur.

Résultat

Le nombre de couleurs or FAILURE.

Exemples

Réduit à 10 le nombre de couleurs utilisées pour l'image butterfly et extrait les régions:

```
pcolorquantization 10 0 examples/butterfly.pan b.pan
pim2rg bpan out.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PColorQuantization( const Img2duc &im_in1, const Img2duc
&im_in2, Img2duc &im_out, int colorNumber, int colorSpace );
```

Auteur: Luc Brun

pcolremoveitem

Suppression d’un élément dans une collection.

Synopsis

pcolremoveitem *name* [*col_in*|-] [*col_out*|-]

Description

L’opérateur **pcolremoveitem** supprime de la collection *col_in* l’élément *name* (s’il existe). La nouvelle collection est retournée dans *col_out*.

Paramètres

- *name* est le nom de l’attribut à effacer.

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE si l’élément n’existe pas.

Exemples

Supprime l’ "foo" de la collection col.pan :

```
pobject2col foo tangram.pan col.pan
pcolremoveitem foo col.pan col.pan
pfile col.pan
```

Voir aussi

Collection

Prototype C++

```
Errc PColRemoveItem( const Collection &col_in_out, const std::string
&name );
```

Auteur: Alexandre Duret-Lutz

pcolrenameitem

Renommage d’un élément dans une collection.

Synopsis

pcolrenameitem *old_name new_name* [*col_in*|-] [*col_out*|-]

Description

L’opérateur **pcolrenameitem** crée la collection *col_out* à partir des éléments de *col_in* en ayant renommé *old_name* en *new_name*.

Paramètres

- *old_name* est le nom de l’élément à renommer.
- *new_name* est le nouveau nom de l’élément.

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Renomme l’élément "foo" en "bar" :

```
pobject2col foo tangram.pan col.pan
pcolrenameitem foo bar col.pan col.pan
pfile col.pan
```

Voir aussi

Collection

Prototype C++

```
Errc PColRenameItem( Collection &col_in_out, const std::string
&old_name, const std::string &new_name );
```

Auteur: Alexandre Duret-Lutz

pcolsetobject

Extraction d'un objet Pandore d'une collection.

Synopsis

pcolsetobject *name* [*col_in*|-] [*obj_in*|-] [*col_out*|-]

Description

L'opérateur **pcolsetobject** extrait l'objet Pandore stocké dans la collection *col_in* sous le nom *name*. Un objet Pandore peut être une image, une carte de région ou un graphe. L'objet est sauvegardé dans le fichier de sortie *obj_out*.

Paramètres

- *name* est le nom de l'objet Pandore dans la collection.

Entrées

- *col_in*: une collection.
- *obj_in*: un fichier Pandore.

Sorties

- *col_out*: un objet Pandore.

Résultat

Retourne SUCCESS ou FAILURE si l'objet nommé *name* n'existe pas ou n'est pas un objet Pandore.

Exemples

Ajoute les images tangram.pan et parrot.pan dans la collection col.pan :

```
pobject2col foo tangram.pan col.pan
pcolsetobject bar col.pan parrot.pan col.pan
pfile col.pan
```

Voir aussi

Collection

Prototype C++

```
Errc PColSetObject( const std::string &name, Collection &col_in_out,
Pobject* &out );
```

Auteur: Alexandre Duret-Lutz

pcolsetvalue

Ajout de la valeur d'un numérique dans une collection.

Synopsis

pcolsetvalue *name type value* [*col_in*|-] [*col_out*|-]

Description

L'opérateur **pcolsetvalue** permet d'ajouter dans la collection *col_in* une valeur numérique *valeur* avec le nom *nom_attribut* le type *type*. Le résultat est dans la collection de sortie *col_out*.

Si l'attribut de nom *nom_attribut* existe déjè, la valeur est remplacée.

Paramètres

- *name* est le nom de la valeur numérique dans la collection d'entrée. C'est une chaîne de caractères sans espace.
- *type* est une type numérique Pandore, parmi [Char, Uchar, Short, Ushort, LOng, Ulong, Float, Double].
- *value* est une valeur numérique dont le domaine dépend du type.

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Ajoute la valeur 10.5 dans la collection col.pan avec le nom "foo" puis vérifie que la valeur est bien dans la collection :

```
pobject2col image tangram.pan col.pan
pcolsetvalue foo Float 10.5 col.pan col.pan
pfile col.pan
pcolgetvalue foo col.pan
pstatus
```

Voir aussi

Collection

Prototype C++

```
Errc PColSetValue( Collection &col_in_out, const std::string &name,
const std::string& type, float value );
```

Auteur: Régis Clouard

pcompactnessselection

Sélection de régions sur leur valeur de compacité.

Synopsis

pcompactnessselection *relation seuil* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pcompactnessselection** permet de sélectionner les régions sur leur valeur de compacité. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

La compacité est un facteur de circularité qui vaut 1 lorsque la région est un cercle, et diminue à mesure que le contour est très découpé ou que la région est allongée.

Elle est calculée par la formule:

```
compacité. (4*PI*surface) / (perimetre*perimetre)
```

Une compacité de 0 correspond à une forme peu compacte.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur réelle [0, 2*]. (* du fait des erreurs de calculs discrets)
  Une compacité proche de 1 (i.e., < 2 du fait des erreurs de calculs discrets) correspond à la compacité d'un cercle.

Entrées

- *rg_in*: une carte de régions

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Exemples

Sélectionne les régions de plus forte compacité :

```
pcompactnessselection 3 0 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PCompactnessSelection( const Reg2d &rg_in,Reg2d &rg_out, int
relation, float seuil );
```

Auteur: Régis Clouard

pcontentsdisplay

Affichage du contenu d'un objet Pandore.

Synopsis

pcontentsdisplay [*-m mask*] [*obj_in*|-]

Description

L'opérateur **pcontentsdisplay** permet d'afficher sur la console le contenu de l'objet *obj_in* dont les valeurs sont non nulles. Grâce au masquage (-m *mask*), il est possible de sélectionner une zone des valeurs à afficher.

Si *obj_in* est une image, **pcontentsdisplay** affiche seulement les valeurs des pixels non nuls.

Si *obj_in* est un graphe, **pcontentsdisplay** affiche la valeur de chacun des noeuds.

Si *obj_in* est une collection, **pcontentsdisplay** affiche la taille en octets de chacun des composants de la collection.

Entrées

- *obj_in*: un objet Pandore.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Affiche le contenu de l'image tangram.pan :

```
pcontentsdisplay examples/tangram.pan
```

- Affiche une partie du contenu de l'image tangram.pan (size 50x50) :

```
pshapedesign 256 256 0 1 50 50 a.pan
pim2rg a.pan m.pan
pcontentsdisplay -m m.pan examples/tangram.pan
```

Voir aussi

Visualisation

Prototype C++

```
Errc PContentsDisplay( const Img2duc &im_in );
```

Auteur: François Angot

pcontourextensionconic

Extension des points terminaux dans la direction du contour par une forme conique.

Synopsis

pcontourextensionconic *longueur* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pcontourextensionconic** consiste à étendre tous les points terminaux des contours de l'image avec un cône dont l'axe est de taille *longueur* dans la direction de poursuite du contour. (*Les contours touchant le bord de l'image ne sont pas étendus*).

L'extension des points terminaux est faite avec un cône de la taille *longueur* dans le sens du contour. Par exemple, le contour 2D sur la gauche est étendu par le contour sur la droite de *longueur*=3 :

```
                    x
                   xx
                  xxx
 xxxx     ->   xxxxxxxx
                  xxx
                   xx
                    x
```

Un contour est une chaîne de pixels non nuls reposant sur un fond nul. Un point du contour est **terminal** lorsqu'il n'a qu'**un seul voisin**. Il peut être alors utile de faire précéder cet opérateur d'opérateurs d'amincissement des contours qui garantissent la 8-connexité (ou la 26-connexité en 3D).

Param&eagrave;tre

- La *longueur* est la longueur en pixels de l'axe du cône à ajouter dans le sens du contour. Une longueur de 0 correspond à aucune extension.

Entrées

- *im_in*: une image d'octets 2D ou 3D.

Sorties

- *im_out*: une image d'octets 2D ou 3D.

Résultat

Le nombre de points terminaux étendus.

Exemples

Ferme les contours obtenus par une simple détection de contours par extension des contours :

```
psobel tangram.pan b.pan
pbinarization 55 1e30 b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
pcontourextensionconic 3 e.pan f.pan
plabeling 8 f.pan out.pan
```

Voir aussi

Contour

Prototype C++

```
Errc PContourExtensionConic( cont Img2duc &im_in, Img2duc &im_out,
int longueur );
```

Auteur: Arnaud Renouf

pcontourextensionrect

Extension des points terminaux dans la direction du contour.

Synopsis

pcontourextensionrect *longueur hauteur* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pcontourextensionrect** consiste à étendre tous les points terminaux des contours de l'image avec un rectangle (ou un parallélogramme en 3D) de taille *longueur* x *hauteur* (x *profondeur*) dans la direction de poursuite du contour. (*Les contours touchant le bord de l'image ne sont pas étendus*).

Un contour est une chaîne de pixels non nuls reposant sur un fond nul. Un point du contour est **terminal** lorsqu'il n'a qu'**un seul voisin**. Il peut être alors utile de faire précéder cet opérateur d'opérateurs d'amincissement des contours qui garantissent la 8-connexité (ou la 26-connexité en 3D).

Paramètres

- *longueur* est la longueur en pixels du rectangle à ajouter dans le sens du contour. Une longueur de 0 correspond à aucune extension.
- *hauteur* est l'épaisseur en pixels du rectangle à ajouter de part et d'autre dans le sens orthogonal au contour. Une hauteur de 0 correspond à aucune extension.

Entrées

- *im_in*: une image de type Uchar.

Sorties

- *im_out*: une image de Uchar.

Résultat

Le nombre de points terminaux étendus.

Exemples

Ferme les contours obtenus par une simple détection de contours par extension des contours :

```
psobel tangram.pan b.pan
pbinarization 60 1e30 b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
pcontourextensionrect 3 3 e.pan f.pan
plabeling 8 f.pan out.pan
```

Voir aussi

Contour

Prototype C++

```
Errc PContourExtensionRect( const Img2duc &im_in, Img2duc &im_out,
int hauteur, int longueur );
```

Auteur: Régis Clouard

pcontourselection

Sélectionner des chaînes de contours isolées sur leur longueur.

Synopsis

pcontourselection *relation longueur* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pcontourselection** consiste à sélectionner dans une image de contours toutes les chaînes de contours isolées ayant sur leur longueur.

La valeur de *longueur* donnée en paramètre et comptée en nombre de pixels.

Une ligne est une chaîne continue en 8 connexité (ou 26 connexité en 3D) de pixels non nuls d'épaisseur 1 pixel, commençant par un point terminal (point que ne possède qu'un seul voisin) et finissant par un point terminal, et qui **ne possède pas d'intersection avec une autre ligne**.

Une courbe fermée et une barbule ne sont pas considérées comme des lignes.

Attention: les points terminaux sont des points qui ne possèdent qu'un voisin. Il peut alors être utile de faire précéder cet opérateur d'opérateurs d'amincissement qui garantissent la 8-connexité (ex: **postamincissement**).

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de longueur.
  - *relation* = 3: contours avec la taille maximale.
  - *relation* = 2: contours ≥ *longueur*.
  - *relation* = 1: contours > *longueur*.
  - *relation* = 0: contours = *longueur*.
  - *relation* = -1: contours < *longueur*.
  - *relation* = -2: contours ≤ *longueur*.
  - *relation* = -3: contours avec la taille minimale.
- La longueur *longueur* est une valeur entière comptée en nombre de pixels.

Entrées

- *im_in*: une image de type Uchar.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Le nombre de lignes supprimées.

Exemples

Sélectionne les contours avec une longueur d'au moins 100 pixels:

```
psobel tangram.pan b.pan
pbinarization 45 1e30 b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
pcontourselection 1 100 e.pan out.pan
pstatus
```

Voir aussi

Contour

Prototype C++

```
Errc PContourSelection( const Img2duc &im_in, Img2duc &im_out, int
relation, int longueur );
```

Auteur: Régis Clouard

pcontrast1quadtree

Segmentation d'une image par quadtree (octree) selon le contraste.

Synopsis

pcontrast1quadtree *seuil* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

pcontrast1quadtree permet de segmenter l'image en différentes régions selon la valeur du contraste. Les régions obtenues sont rectangulaires.

Le principe de l'algorithme est le suivant:

- Si un bloc n'est pas homogéne (i.e. le contraste est supérieur au seuil) alors on divise le bloc en 4 blocs égaux et on réapplique l'algorithme sur chacun des blocs.

On utilise ici la valeur du contraste intérieur calculé par:

```
contraste(R)= 1/N * sum(max(C(s,t), t in V(s) et t in R))
avec C(s,t)= | im_in[s] - im_in[t] |
où N = nombre de pixels de la région.
```

En 3D, le résultat est un octree, c'est à dire une carte de régions composée de cubes.

Paramètres

- *seuil* est la variation maximum en niveaux de gris pour qu'une région soit acceptée comme uniforme. Les valeurs appartiennent à l'intervalle des valeurs de niveaux de gris possible de l'image *im_in*.

Entrées

- *im_in*: une image en niveaux de gris.

Sorties

- *rg_out*: une carte de régions de la dimension de l'image d'entrée.

Résultat

Retourne le nombre de régions obtenues.

Exemples

Construit une partition de l’image tangram.pan:

```
pcontrast1quadtree 10 tangram.pan a.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PContrast1Quadtree( const Img2duc &im_in, Reg2d &rg_out, Uchar seuil );
```

Auteur: Laurent Quesnel

pcontrast1value

Calcul du contraste global d'une image ou d'un graphe.

Synopsis

pcontrast1value [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pcontrast1value** permet de calculer le contraste total d'une image ou des valeurs des sommets du graphe.

La mesure de contraste est faite selon la formule:

```
C= SOMME(max (C(s,t), C in V(s))) / N
et C(s,t) = |im_in[s] - im_in[t]| / (K-1)
où K est le nombre de niveaux de gris possible,
et N le nombre de pixels de l'image.
```

Pour les graphes, le contraste est calculé sur les valeurs de noeud.

Les valeurs de contraste de chaque bande sont stockées dans la collection *col_out*.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *col_out*: une collection de valeurs réelles.

Résultat

Retourne le contraste global (pour la première bande uniquement).
Cette valeur peut être récupérée par l'opérateur **pstatus**.

Exemples

Mesure le contraste global de l'image tangram.pan (version Unix):

```
pcontrast1value tangram.pan col.pan
var=‘pstatus‘
echo "Contraste = $val"
```

Mesure le contraste global de l’imge tangram.pan (version MsDos):

```
pcontrast1value tangram.pan col.pan
call pstatus
call pset var
echo Contraste = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Float PContrast1Value( const Img2duc &im_in, Collection & col_out );
```

Auteur: Régis Clouard

pcontrastaggregation

Croissance des régions d'une carte selon le contraste intérieur.

Synopsis

pcontrastaggregation *connexite seuil* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pcontrastaggregation** consiste à agglomérer des pixels à une région connexe lorsque la valeur du pixel est proche de celle de la région, c'est à dire quand sa valeur appartient:

```
[m(R) - contraste(R), m(R) + contraste(R)],
```

Les pixels à agglomérer sont les pixels non encore étiquétés dans la carte de régions *rg_in* (ceux qui ont un label=0).

Le contraste est estimé ici par:

```
contraste(R)=max(R)-min(R).
```

On agglomère un pixel à une région R connexe si:

```
|contraste(R)-contraste(R+im_in[p])| <= seuil
```

Le contraste des régions de *rg_in* n'est pas recalculé pour éviter de trop s'éloigner de la situation initiale. On préferera des exécutions itératives de cet opérateur. On pourra par exemple itérer cet opérateur jusqu'à ce que le résultat de pstatus = 0. Ainsi, à chaque appel de l'opérateur le contraste est recalculé avec les nouvelles régions.

La carte de sortie *rg_out* a le même nombre de labels que la carte d'entrée *rg_in*.

Paramètres

- *connexite* définit la relation de voisinage entre pixel : 4 ou 8 pour le 2D, et 6 ou 26 pour le 3D.
- *seuil* fixe l'écart toléré au contraste d'une région pour y agglomérer un pixel. C'est une valeur de l'intervalle [0..niveau de gris].

Entrées

- *rg_in*: une carte de régions.
- *im_in*: une image.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre total de pixels qui ont été aggrégés à une région. Retourne FAILURE en cas de problème.

Exemples

Aggrège les pixels des pièces de tangram :

```
pbinarization 96 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pcontrastaggregation 8 20 b.pan tangram.pan out.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PContrastAggregation( const Reg2d &rg_in, const Img2duc &im_in,
Reg2d &rg_out, int connexite, Uchar seuil );
```

Auteur: Régis Clouard

pcontastbinarization

Multi-seuillage de l'image par analyse du contraste aux frontières.

Synopsis

pcontrastbinarization *nbclass* [*-m mask*] [*im_in*|-] [*im_amp*|-] [*im_out*|-]

Description

L'opérateur **pcontrastbinarization** permet de seuiller l'image initiale *im_in* par une méthode basée sur l'analyse de l'histogramme des amplitudes de gradient le long des frontières données dans *im_amp*.

Cet opérateur est basé sur l'algorithme de Kohler:
soit p et q deux pixels voisins de niveaux de gris respectifs p(x,y) et q(x,y). Un contour entre p et q est détecté par un seuil `s` si et seulement si:

```
p(x,y) <= s <= q(x,y) ou q(x,y) <= s <= p(x,y).
```

L'ensemble de contours détectés par `s` est :

```
K(s)={ paires(p,q) / p et q voisins et p(x,y)<=s<=q(x,y) ou q(x,y)<=s<=p(x,y) }
```

Le contraste total des contours détectés par `s` est donné par :

```
C(s) = sum(min(abs(s-p(x,y)),abs(s-q(x,y)))
```

la somme étant faite sur tous les éléments (p,q) de K(s).
Le contraste moyen est:

```
Cm(s) = C(s) / card(K(s))
```

Le seuil est pris comme le maximum de la fonction histogramme.

Entrées

- *im_in*: une image de niveaux de gris en octets (Img2duc ou Img3duc);
- *im_amp*: une image d'amplitude de gradient en niveaux de gris.

Sorties

- *im_out*: une image.

Résultat

Retourne la valeur de seuil.

Exemples

Segmente l’image tangram:

```
pgradient 1 tangram.pan a.pan b.pan
pnonmaximasuppression a.pan b.pan c.pan
pbinarization 10 1e30 c.pan d.pan
pcontrastbinarization tangram.pan d.pan out.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PContrastBinarization( const Img2duc &im_in1, Img2duc &im_in2,
Img2duc &im_out );
```

Référence

R. Kohler, "A segmentation system based on thresholding", *CGIP*, No. 15, pp 319-338, 1981.

Auteur: Régis Clouard

pcontrastmerging

Fusion prioritaire de régions selon le critère du contraste.

Synopsis

pcontrastmerging *nb_fusion seuil* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

L'opérateur **pcontrastmerging** permet de fusionner les régions de la carte de régions *rg_in* selon le critère du contraste.

La notion de voisinage entre les régions est détenue par le graphe *gr_in*.

Le principe de l'algorithme est le suivant:
Pour chaque région de la carte de régions, on calcule la différence de contraste intérieur avec chacune de ses voisines. Si la différence est inférieure au *seuil* donné en paramètre, alors les régions sont fusionnées.

On utilise ici l'algorithme de croissance prioritaire qui consiste à fusionner à chaque fois les 2 régions dont la différence est la plus faible.

Le contraste est calculé par:

```
contraste(R)= max(R) - min(R).
```

Paramètres

- *nb_fusion* permet de spécifier le nombre de fusion à effectuer (la valeur -1 signifie d'ignorer ce paramètre et d'exécuter l'algorithme tant qu'il y a des fusions possibles).
- *seuil* permet de spécifier la tolérance maximale sur l'écart de contraste entre 2 régions. Les valeurs appartiennent à l'intervalle [0..nombre de niveaux de gris].

Entrées

- *rg_in*: une carte de régions.
- *gr_in*: un graphe.
- *im_in*: une image.

Sorties

- *rg_out*: une carte de régions.
- *gr_out*: un graphe.

Résultat

Retourne le nombre de fusions effectuées.

Exemples

Fusionne les régions issues d'une partition de l'image tangram.pan :

```
puniformityquadtree 0.9 tangram.pan a.pan
prg2gr a.pan b.pan
pcontrastmerging -1 45 a.pan b.pan tangram.pan c.pan d.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PContrasteMerging ( const Reg2d &rg_in, const Graph2d &gr_in,
const Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, int nb_fusion,
Uchar seuil);
```

Auteur: Laurent Quesnel

pcontrastquadtree

Segmentation d'une image par quadtree (octree) selon le contraste.

Synopsis

pcontrastquadtree *seuil* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pcontrastquadtree** permet de segmenter l'image en différentes régions selon la valeur du contraste. Les régions obtenues sont rectangulaires.

Le principe de l'algorithme est le suivant:

- Si un bloc n'est pas homogène (i.e. le contraste est supérieur au seuil) alors on divise le bloc en 4 blocs égaux et on réapplique l'algorithme sur chacun des blocs.

On utilise ici la valeur du contraste calculée par:

```
contraste(R)= max(R) - min(R).
```

En 3D, le résultat est un octree, c'est à dire une carte de régions composée de cubes.

Paramètres

- *seuil* est la valeur de contraste maximum pour qu'une région soit acceptée comme uniforme. Les valeurs appartiennent à l'intervalle des niveaux de gris possibles pour l'image.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *rg_out*: une carte de régions de la dimension de l'image d'entrée.

Résultat

Retourne le nombre de régions obtenues.

Exemples

Construit une partition de l’image tangram.pan:

```
pcontrastquadtree 10 tangram.pan a.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PContrastQuadtree( const Img2duc &im_in, Reg2d &rg_out, Uchar
seuil );
```

Auteur: Laurent Quesnel

pcontastthresholding

Multi-seuillage de l'image par analyse du contraste aux frontières.

Synopsis

pcontrastthresholding *nbclass* [*-m mask*] [*im_in*|-] [*im_amp*|-] [*im_out*|-]

Description

L'opérateur **pcontrastthresholding** permet de multiseuiller l'image initiale *im_in* par une méthode basée sur l'analyse de l'histogramme des amplitudes de gradient le long des frontières données dans *im_amp*.

Cet opérateur est basé sur l'algorithme de Kohler:
soit x et y deux pixels voisins de niveau de gris i et j. Un contour entre x et y est détecté par un seuil s si et seulement si:

```
i<=s<=j ou j<=s<=i.
```

L'ensemble de contours détectés par s est:

```
K(s)={ paires(x,y) / x et y voisins et i<=s<=j ou j<=s<=i }
```

Le contraste total des contours détectés par s est donné par:

```
C(s) = sum(min(abs(s-i),abs(s-j))
```

la somme étant faite sur tous les éléments (x,y) de K(s).
Le contraste moyen est:

```
Cm(s) = C(s) / card(K(s))
```

Les seuils sont pris comme les maxima de la fonction histogramme. Seuls sont conservés les *nbclass*-1 plus grands maxima.

L'image de sortie *im_out* est contruite avec les seuils détectés, telle que:

```
im_out[y][x]=seuil[k] si seuil[k-1]<im_out[y][x]<=seuil[k].
```

Le dernier seuil est égal à la valeur maximale 255.

Paramètres

- *nbclass* est le nombre de classes maximum exigée. sortie.

Entrées

- *im_in*: une image de niveaux de gris en octets (Img2duc ou Img3duc);
- *im_amp*: une image d'amplitude de gradient en niveaux de gris.

Sorties

- *im_out*: une image.

Résultat

Retourne le nombre de classes de sortie.

Exemples

Segmente l'image tangram:

```
pgradient 1 tangram.pan a.pan b.pan
pnonmaximasuppression a.pan b.pan c.pan
pthresholding 10 1e30 c.pan d.pan
pcontrastthresholding 2 tangram.pan d.pan out.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PContrastThresholding( const Img2duc &im_in1, Img2duc &im_in2,
Img2duc &im_out, int nbclass );
```

Référence

R. Kohler, "A segmentation system based on thresholding", *CGIP*, No. 15, pp 319-338, 1981.

Auteur: Régis Clouard

pcontrastvalue

Calcul du contraste global d'une image ou d'un graphe.

Synopsis

pcontrastvalue [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pcontrastvalue** permet de calculer le contraste total d'une image ou des valeurs des sommets du graphe.

La mesure de contraste est faite selon la formule:

```
contraste(im_in) = max(im_in) - min(im_in).
```

Pour les graphes, le contraste est calculé sur les valeurs de noeud.

Les valeurs de contraste de chaque bande sont stockées dans la collection *col_out*.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *col_out*: une collection de valeurs réelles.

Résultat

Retourne le contraste global (pour la première bande uniquement).
Cette valeur peut être récupérée par l'opérateur **pstatus**.

Exemples

Mesure le contraste global de l'image tangram.pan (version Unix):

```
pcontrastvalue tangram.pan col.pan
var=‘pstatus‘
echo "Contraste = $val"
```

Mesure le contraste global de l'image tangram.pan (version MsDos):

```
pcontrastvalue tangram.pan col.pan
call pstatus
call pset var
echo Contraste = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Float PContrastValue( const Img2duc &im_in, Collection & col_out );
```

Auteur: Régis Clouard

pconvexhull

Calcul de l'enveloppe convexe des régions.

Synopsis

pconvexhull [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pconvexhull** reconstruit la forme des régions à partir de leur enveloppe convexe. Le numéro des régions de *rg_in* est conservé dans *rg_out*.

Attention: il peut y a voir chevauchement des enveloppes convexes. Dans ce cas, c'est la région ayant le numéro de label le plus élevé qui se trouve au dessus des autres.

Entrées

- *rg_in*: une carte de régions.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Dessine l'enveloppe convexe autour des régions de la carte rin.pan :

```
pcconvexhull rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PConvexHull( const Reg2d &, Reg2d &im_out );
```

Auteur: Régis Clouard

pconvexityselection

Sélection de régions sur leur valeur de convexité.

Synopsis

pconvexityselection *relation seuil* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pconvexityselection** permet de sélectionner les régions sur leur valeur de convexité. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

Le calcul de la mesure de convexité est fait par :

```
convexité. Surface de la région / Surface de l'enveloppe convexe.
```

Une région fortement convexe (ie, qui épouse parfaitement son enveloppe convexe) a une valeur de convexité. 1.
Une région faiblement convexe a une valeur de convexité << 1.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur réelle de l'intervalle [0,1] où 1 correspond à une région parfaitement convexe.

Entrées

- *rg_in*: une carte de régions

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Exemples

Sélectionne les régions avec un degré de convexité ≥ 0.5 :

```
pconvexityselection 2 0.5 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PConvexitySelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, Ushort seuil );
```

Auteur: Régis Clouard

pconvolution

Convolution d'une image par un noyau.

Synopsis

pconvolution *filename* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pconvolution** permet de convoluer l'image initiale *im_in* par le noyau donné dans le fichier *filename*. La convolution consiste en l'opération:

```
im_out[y][x]= Sum_k,l { im_in[y+l][x+k]* mask[l][k] } / norme;
```

Le résultat est normalisé par la somme des valeurs des coefficients (`norme`) ou par 1 si la somme est nulle.

Le fichier *filename* est un fichier texte qui contient plusieurs lignes formées comme suit: La taille du noyau est donné en premier sous la forme: nombre_plan*nombre_ligne*nombre_colonnes, puis les coefficients (flottant ou entiers) sont placés en séquence séparés par un espace. L'ordre est celui des plans, puis lignes puis colonnes.

Soit pour un noyau 3D:

```
nprof*nlig*ncol
c1 c2 c3 ...
```

Par exemple, le fichier suivant correspond au noyau d'approximation du Laplacien 2D:

```
3*3
-1.0 -1.0 -1.0 -1.0 8 -1 -1 -1 -1
```

L'image de sortie est une image de Float.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de Float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Le noyau suivant peut être utilisé pour appliquer un filtre moyenneur:

```
kernel.txt:
3*3
1 1 1 1 1 1 1 1 1

pconvolution kernel.txt tangram.pan a.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PConvolution( const Img2duc &im_in1, Img2dsf &im_in2, char*
filename );
```

Auteur: Régis Clouard

pcopyborder

Copie les valeurs du bord d'une image dans une autre.

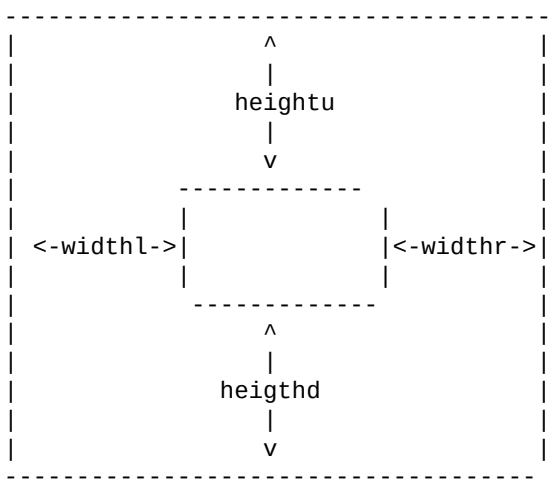
Synopsis

pcopyborder *widthl widthr heightu heightd depthf depthb* [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pcopyborder** permet de copier les pixels du bords de l'image *im_in2* dans l'image *im_in1*. L'image de sortie *im_out* est donc une copie de l'image *im_in1* sauf le bord qui est une copie de l'image *im_in2*. La bordure d'une image 3D à une taille de largeur *widthl* pixels à gauche et *widthr* pixels à droite, de hauteur *heightu* pixels en haut et *heightd* pixels en bas et de profondeur *depthf* pixels devant et *depthb* pixels derrière.

Pour une image 2D, les dimensions sont:



Pour une image couleur ou multispectrale, la bordure est modifée avec la même valeur *val* sur toutes les bandes.

Paramètres

- *depthf*, *depthb*, *heightu*, *heightd*, *widthl*, *widthr* donnent la taille en pixels de chacune des trois bordures en profondeur, en hauteur et en largeur.

  Dans le cas 2D, *depthf et depthb* ne sont pas utilisés mais doivent être donnés.

Entrées

- *im_in1*: une image.
- *im_in2*: une image de même type que *im_in1*.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Sélection des objets d'une sous image de tangram.pan qui ne touchent pas le bord de l'image.

```
pextractsubimage  30 30 0 150 150 0 tangram.pan in.pan
pcopycst 0 ii.pan i1.pan
pcopyborder 1 1 1 1 1 1 i1.pan in.pan i2.pan
pdilationreconstruction 8 i2.pan ii.pan i3.pan
pdif in.pan i3.pan out.pan
```

Voir aussi

Utilitaire

Prototype C++

```
Errc PCopyBorder( const Img3duc &im_in1, const Img3duc &im_in2,
Img3duc &im_out, Long widthl, Long widthr, Long heightu, Long
heigthd, Long depthf, Long depthb);
```

Auteur: Régis Clouard

pcorrelationbinarization

Binarisation de l'image par maximisation de la corrélation interclasse.

Synopsis

pcorrelationbinarization [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pcorrelationbinarization** classifie les pixels de l'image d'entrée *im_in* en 2 deux classes. La valeur de seuil est déterminée comme la valeur de niveau de gris `s` qui maximise les valeurs de correlation du fond et des objets pris séparément. La quantité totale de correlation pour un seuil s est donnée par :

```
TC(s) = Cb(s) + Cf(s) { correlation pour le fond + correlation pour les objets }
      = -ln[G(s)*G'(s)] +2*ln[P(s)*(1-P(s))]
  où P(s) = SUM{i=0->s} [p(i)]
  et G(s) = SUM{i=0->s} [p(i)^2)]
  et G'(s) = SUM{i=s->m-1} [(p(i)^2]
  et pi = fi/W*H
```

Le critère de correlation se détermine à partir du seuil smax tel que:

```
TC(smax) = max TC(s)
```

Entrées

- *im_in*: une image niveaux de gris de Uchar (Img2duc, Img3duc).

Sorties

- *im_out*: une image niveaux de gris de Uchar (Img2duc, Img3duc).

Résultat

Retourne la valeur de seuil détectée.

Exemples

Segmentation des pièces de tangram :

```
pcorrelationbinarization tangram.pan a.pan
```

Voir aussi

Thresholding

Prototype C++

```
Errc PCorrelationBinarization( const Img2duc &im_in, Img2duc &im_out
);
```

Référence

J-C Yen, F-J Chang, S. Chang, "A New Criterion for Automatic Multilevel Thresholding", *IEEE Trans. on Image Processing*, vol. 4, no. 3, pp 370-378, 1995.

Auteur: Régis Clouard

pcorrelationcoefficient

Calcul de la valeur de corrélation entre deux vecteurs de variables.

Synopsis

pcorrelationcoefficient *name_in1 name_in2 name_out* [*col_in1*|-] [*col_in2*|-] [*col_out*|-]

Description

L'opérateur **pcorrelationcoefficient** calcule la valeur de corrélation entre deux variables représentées chacune par un vecteur de valeurs stockés dans une collection.

Le coefficient de correlation entre les deux vecteurs X et Y est donné par :

```
r_{xy} = \frac{\sum\limits_{i=1}^n (x_i-\bar{x})(y_i-\bar{y})}{(n-1) s_x s_y}
       = \frac{\sum\limits_{i=1}^n (x_i-\bar{x})(y_i-\bar{y})} {\sqrt{\sum\limits_{i=1}^n (x_i-\bar{x})^2 \sum\limits_{i=1}^n (y_i-\bar{y})^2}},
```

où x et y sont les moyennes de X et de Y, et sx et sy sont les écart-types de X et Y.

Paramètres

- name_in1, name_in2: la collection *col_in* contient les vecteurs de caractéristiques à partir desquels il faut faire les calculs. S'il y a `n` vecteurs de `p` caractéristiques chacun, la collection doit contenir `p` tableaux `name_in.1`, `name_in.2`, ..., `name_in.p` de `n` flottants chacun.
- name_out: la collection *col_out* contient en sortie :
  - `name_out.mat` : le tableau des `p*p` s de la matrice de covariance ;
  - `name_out.inv` : le tableau des `p*p` valeurs de l'inverse de la matrice de covariance ;
  - `name_out.det` : le déterminant de la matrice de covariance
  - `name_out.det` : le tableau des `p` valeurs du vecteur de la moyenne des caractéristiques.

Entrées

- *col_in*: une collection contenant le tableau de valeurs name_in1.
- *col_in*: une collection contenant le tableau de valeurs name_in2 de même taille que le tableau name_in1.

Sorties

- *col_out*: une collection avec la valeur de sortie.

Résultat

Retourne la valeur de correlation.

Exemples

```
pcorrelationcoefficient Correctness Correctness r ccorrectness1.pan ccorrectness2.pan toto.pan
```

Voir aussi

Vecteur

Prototype C++

```
Errc PCorrelationCoefficient( const Collection &col_in1, const
Collection &col_in2, Collection &col_out, const string &name_in1,
const string &name_in2, const string &name_out );
```

Auteur: Régis Clouard

pcreatearray

Création d'une collection contenant un vecteur vierge.

Synopsis

pcreatearray *name type size value* [*col_out*|-]

Description

L'opérateur **pcreatearray** crée une collection *col_out* contenant un vecteur nommé *name* de *size* valeurs de types *type*. Le vecteur *name* est initialisé avec la valeur *value*.

Paramètres

- *type* est le nom de l'un des types de base (`Uchar, Short, Double` ...).
- *size* est la taille du vecteur.
- *value* est une valeur du *type* donné.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Crée un vecteur de 128 float de valeur 10.5:

```
pcreatearray foo float 128 10.5 col.pan
pcol2txt col.pan
```

Voir aussi

Vecteur

Prototypes C++

```
Errc PCreateArray( const Collection &cd, Long size, Uchar value,
std::string name );
```

Auteur: Alexandre Duret-Lutz

pcrosscorrelation

Correlation d'une image par un noyau.

Synopsis

pcrosscorrelation [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pcrosscorrelation** permet de mesurer le degré de correspondance entre chaque partie de l'image *im_in1* avec le motif *im_in2*. La valeur de corrélation appartient à l'intervalle [0,1] où 1 est la valeur de correspondance maximale.

Entrées

- *im_in1*: une image en niveaux de gris (Img2duc).
- *im_in2*: une image d'un motif en niveau de gris.

Sorties

- *im_out*: une image de réels entre 0 et 1.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Détection des occurrences du motif letter.pan dans l'image page.pan.

```
pcrosscorrelation page.pan letter.pan image1.pan
```

Voir aussi

Template Matching

Prototype C++

```
Errc PCrossCorrelation( const Img2duc &im_in1, const Img2duc
&im_in2, Img2dsf &im_out );
```

Auteur: Régis Clouard

pdelaunay

Construction du graphe de Delaunay discret.

Synopsis

pdelaunay [*-m mask*] [*rg_in*|-] [*gr_out*|-]

Description

L'opérateur **pdelaunay** construit le graphe de Delaunay. Le graphe de delaunay est le dual du diagramme de Voronoï.

On appelle polygone de Voronoï associé du site Pi la région Vor(Pi) (chaque région étant l'ensemble de points (x,y) les plus proches à un point de P) telle que chaque point de P a pour plus proche site Pi.

Les germes donnés dans la carte *rg_in* sont les sites du diagramme à construire. Ils forment les sommets du graphe de sortie *gr_out*. **Un germe est une région de taille 1 et de label unique**. Si les germes ne sont ni des régions de taille 1 pixel ni de label unique, une erreur est retournée.

Le graphe de Delaunay est construit en reliant par un arc toutes les paires de sites dont les régions de Voronoï correspondantes sont adjacentes, c'est à dire séparées par une arête de Voronoï.

Les coordonnées du germe forment les coordonnées du sommet.

La valeur de label du germe est utilisée pour le numéro du sommet dans le graphe.

Chaque arc est pondéré avec la valeur 1.0.

Entrées

- *rg_in*: une carte de régions (contenant les germes étiquetés).

Sorties

- *gr_out*: un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule le graphe de Delaunay à partir des centres de gravité des objets dans l'image tangram.pan.

```
pbinarization 90 1e30 tangram.pan a.pan
plabeling 8 a.pan r1.pan
pcenterofmass r1.pan r2.pan
pdelaunay r2.pan g.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PDelaunay( const Reg2d &rg_in, Graph2d &gr_out );
```

Auteur: Sébastien Bougleux

pdenoisePDE

Régularisation d'images multivaluées par lissage anisotrope basé EDP.

Synopsis

pdenoisePDE *nb_iter amplitude sharpness anisotropy alpha sigma* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pdenoisePDE** permet de régulariser une image couleur ou multi-valuée 2D ou 3D. La technique de régularisation utilisée permet de traiter de manière efficace les artefacts locaux rencontrés dans des images (bruit ou artefacts de compression par exemple). La régularisation est anisotrope et préserve les courbures, c'est-è-dire qu'elle permet de lisser l'image tout en préservant les structures importantes des images (bords, coins, discontinuités).

Le temps d'exécution peut être très long selon les valeurs de paramètre passées.

Paramètres

- *nb_iter* définit le nombre d'itérations d'EDP effectués. Beaucoup d'itérations permettent de lisser l'image de manière plus importante. Pour des images peu bruitées, une valeur de 1 convient.
- *amplitude* définit l'amplitude du lissage lors d'une itération (pas de temps de l'EDP). Plus ce paramètre est important, plus l'image va se lisser rapidement à chaque itération. En général, une valeur entre 5 et 200 convient.
- *sharpness* définit une valeur de contraste de référence des contours à préserver. Plus ce paramètre est important, plus les contours seront préservés (mais éventuellement le bruit aussi!). Une valeur de 0 indique que chaque pixel sera lissé avec la même force, éventuellement dans des directions différentes. Une valeur entre 0 et 2 convient généralement.
- *anisotropy* définit le degré d'anisotropie du lissage. Un lissage très anisotrope (anisotropy=1) est très fortement orienté dans les directions des contours. Un lissage isotrope au contraire, ne favorise aucune direction de lissage (anisotropy=0). Selon le type de bruit, il est utile de limiter l'anisotropie pour éviter un effet d'apparition de textures. Ce paramètre doit prendre sa valeur entre 0 et 1.
- *alpha* définit la variance estimée du bruit (par exemple 0.1).
- *sigma* définit un paramètre d'échelle de la géométrie de l'image. Avant chaque itération, la géométrie locale de l'image est évaluée. Le paramètre sigma correspond au pré-lissage de cette géométrie. Plus sigma est grand, moins les détails seront préservés, mais plus le lissage semblera cohérent. Plus sigma est petit, plus les détails seront préservés, mais en cas de bruit, le lissage peut être incohérent. En général, une valeur de sigma inférieur à 1 est suffisante (par exemple 0.8).

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Réduit le bruit de l'image "tangram.pan" en seulement 2 itérations

```
pdenoisePDE 2 100 2 0.7 0.1 0.8 tangram.pan a.pan
visu a.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PDenoisePDE( const Imx3d &ims, Imx3d &imd, Long nb_iter, Float amplitude, Float sharpness, Float anisotropy, Float alpha, Float sigma );
```

Reference

D. Tschumperlé, "*Fast Anisotropic Smoothing of Multi-Valued Images using Curvature-Preserving PDE's*", Cahier du GREYC No 05/01, Avril 2005.

Avertissement

Auteur: D. Tschumperlé

pdensityselection

Sélection de régions sur leur valeur de densité.

Synopsis

pdensityselection *relation seuil* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pdensityselection** permet de sélectionner les régions sur leur degré de densité. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

La densité est le rapport entre le nombre de pixels contenu dans la région et donnéés dans *im_in* et la surface de la région donnée dans *rg_in*.

```
densite = nombre de pixels/surface de la région.
```

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur réelle [0..1] qui correspond au degré de sphéricité.

Entrées

- *rg_in*: une carte de régions 2D.
- *im_in*: une image 2d d'octets.

Sorties

- *rg_out*: une carte de régions 2D.

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Exemples

Sélectionne les régions avec le plus fort facteur de densité:

```
pdensityselection 3 0 a.pan tangram.pan b.pan
```

Voir aussi

Région

Prototype C++

```
Errc PDensitySelection( const Reg2d &rg_in, const Img2duc &ims,
Reg2d &rg_out, int relation, float seuil );
```

Auteur: Régis Clouard

pdepth2graylevel

Construction d'une image de niveau de gris 2D à partir d'une image 3D.

Synopsis

pdepth2graylevel *threshold* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pdepth2graylevel** construit une image 2D à partir d'une image 3D, où les profondeurs de l'image 3D *im_in* sont converties en valeur de niveaux de gris dans l'image 2D *im_out*. La profondeur est définie comme le plan du premier pixel ayant une valeur > *threshold*. Le premier plan est supposé être à la profondeur 0.

L'algorithme est le suivant:

```
for (z=0; z< depth(im_in); z++)
    if (im_in[z][y][x] > threshold ) then im-out[p.y][p.x] = p.z;
```

Paramètres

- *threshold* définit la couleur maximum de transparence. Chaque pixel de valeur ≤ *threshold* est considérée comme une couleur transparente.

Entrées

- *im_in*: une image de niveaux de gris 3D de Uchar.

Sorties

- *im_out*: une image 2D de Slong (Img2dsl).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit un Random Dot Stereogram à partir de l'image cyto3d.pan:

```
pdepth2graylevel 50 cyto3d.pan i0.pan
pmultcst 10 i0.pan i1.pan
prds i1.pan rds_out.pan
```

Voir aussi

Utilitaire, pgraylevel2depth

Prototype C++

```
Errc PDepth2Graylevel( const Img3duc &im_in, Img2dsl &im_out, long
threshold );
```

Auteur: Jean-Marie Janik

pderavi

Multiseuillage de l'image par analyse de la matrice de co-occurences selon Deravi.

Synopsis

pderavi *length* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pderavi** permet de multiseuiller l'image initiale *im_in* par classification des pixels selon l'algorithme de Deravi.

Cet algorithme est basé sur le calcul de la matrice de co-occurence Tkl non symétrique défini avec les voisins 0 et 6. Pour chaque niveau de gris n de [0..N-1] on calcule la probabilité conditionnelle de transition P(n) entre deux régions séparées par le niveau de gris n par:

```
- P1(n) = [Somme(Somme((Tkl)*(Tkl))] / [Somme(Somme(Tkl))+ Somme(Somme(Tpq))]
```

avec k=[0..n], l=[n+1..N-1] et p=[0..n], q=[0..n]

```
- P2(n) = [Somme(Somme((Tkl)*(Tkl))] / [Somme(Somme(Tkl))+ Somme(Somme(Tpq))]
```

avec k=[n+1..N-1], l=[0..n] et p=[n+1..N-1], q=[n+1..N-1]

```
- P(n) = (P1(n) + P2(n) ) / 2;
```

La recherche des minima locaux de P(n) se fait sur une plage de *length* niveaux de gris de part et d'autre du niveau de gris n.

Remarque: Cet opérateur ne fonctionne que sur des images de Char parce qu'il faut que les transitions Tkl soient significatives (ie, nombre de (k,l) restreints). Il faut donc s'arranger pour transformer les autres types d'images en image de Uchar.

L'image de sortie *im_out* est contruite avec les seuils détectés, telle que:

```
im_out[y][x]=seuil[k] si seuil[k-1]<im_out[y][x]<=seuil[k].
```

Le dernier seuil est égal à la valeur maximale 255.

Paramètres

- *length* définit la plage de recherche des minima de la fonction énergie. Il est définit en niveaux de gris. Plus la valeur est grande, moins il y a de classes en sortie.

Entrées

- *im_in*: une image d'octets (Img2duc, Img3duc).

Sorties

- *im_out*: une image d'octets (Img2duc, Img3duc).

Résultat

Retourne le nombre de seuil détectés.

Exemples

Segmente l'image tangram.pan et affiche le nombre de classes detectées:

```
pderavi 15 tangram.pan out.pan
pstatus
```

Voir aussi

Seuillage

Prototype C++

```
Errc PDeravi( const Img2duc &im_in, Img2duc &im_out, int length );
```

Référence

F. Deravi et al., "Gray level thresholding using second-order statistics", *Pattern Recognition Letter*, Vol. 1, No. 5-6, pp. 417-422, 1983.

Auteur: Régis Clouard

pderiche

Détection et localisation des contours par l'algorithme de Deriche.

Synopsis

pderiche *sigma* [*-m mask*] [*im_in*|-] y [*im_amp*|-] [*im_dir*|-]

Description

L'opérateur **pderiche** permet de localiser les contours de l'image *im_in*. L'image de sortie *im_amp* est construite avec les valeurs d'amplitude maximales dans la direction du gradient. Le reste est mis à 0.

La valeur d'amplitude du gradient en un point reflète la variation de niveau de gris observée dans l'image *im_in* en ce point. Plus cette valeur est élévée plus cette variation est forte.

L'extraction se fait en trois étapes :

1. lissage ;
2. calcul du gradient en chaque point de l'image ;
3. extraction des maxima locaux avec ajustement des niveaux.

La direction du gradient est obtenue par arctan(dy/dx) mesurés en radians. L'image de direction *im_dir* est nécessairement de type float (valeurs dans [0,2*PI]).
Attention: la direction suit le repère image, c'est-à-inversée par rapport au système trigonométrique habituel puisque les ordonnées sont dirigées vers la bas.

Remarque: Le bord de l'image de sortie *im_out* de taille 1 pixel est mis à 0.

Paramètres

- *sigma* donne l'intensité du lissage associé à la détection du gradient. Les valeurs sont typiquement dans l'intervalle [0..10]. Plus la valeur est faible, plus le lissage est fort et donc moins il y aura de contours. Une valeur de 0 correspond à un lissage total de l'image (donc il ne reste plus de gradient). Une valeur typique est 1.

Entrées

- *im_in*: une image.

Sorties

- *im_amp*: une image avec l'amplitude du gradient du même type que l'image im_in.
- *im_dir*: une image avec les directions du gradient de type float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Détection de contours dans l'image tangram.pan:

```
pderiche 3 tangram.pan a.pan b.pan
pbinarization 10 1e30 a.pan c.pan
```

Voir aussi

Détection de contours

Prototype C++

```
Errc PDeriche( const Img2duc &im_in, Img2duc &im_out, Img2duc
&im_dir, float sigma );
```

Auteur: Carlotti & Joguet

pderichesmoothing

Lissage de Deriche.

Synopsis

pderichesmoothing *alpha* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pderichesmoothing** permet de lisser une image par la méthode de Deriche.

Paramètres

- *alpha* donne l'intensité du lissage associé à la détection du gradient. Les valeurs sont typiquement dans l'intervalle [0..10]. Plus la valeur est faible, plus le lissage est fort. Une valeur de 0 correspond à un lissage total de l'image (donc il ne reste plus de gradient). Une valeur typique est 1.

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applique le lissage de Deriche sur l'image tangram.pan:

```
pderichesmoothing 1 tangram.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PDericheSmoothing( const Img2duc &im_in, Img2duc &im_out,
double alpha );
```

Auteur: Carlotti & Joguet

pdif

Différence d'images ou de graphes et différence symétrique entre cartes de régions.

Synopsis

pdif [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pdif** calcule la différence symétrique entre l'image *im_in1* et l'image *im_in2*.

La formule de calcul est la suivante :

```
pixel(im_out) = ABS((pixel(im_in1) - pixel(im_in2))
```

Les deux images d'entrée *im_in1* ou *im_in2* doivent être de même type. Au besoin, il est nécessaire d'utiliser les opérateurs de coercition. L'image de sortie est aussi de même type que les images d'entrée.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Pour les cartes de régions, l'opérateur **pdif** correspond à la différence symétrique :

Union(*im_in1*,*im_in2*) - Intersection(*im_in1*,*im_in2*).

La carte de régions de sortie contient une nouvelle numérotation des labels.

Entrées

- *im_in1*: une image, un graphe ou une carte de régions
- *im_in2*: une image, un graphe ou une carte de régions

Sorties

- *im_out*: un objet du même type que im_in1 and im_in2.

Résultat

Retourne SUCCESS ou FAILURE.

Pour les cartes de régions, retourne la valeur de label maximale.

Exemples

```
pdif a.pan b.pan c.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PDif( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

pdilation

Dilatation des points de fort contraste d'une image.

Synopsis

pdilation *num_se halfsize* [*-m mask*][*im_in*|-][*im_out*|-]

Description

L'opérateur **pdilation** permet de dilater les points de plus fort contraste selon un élément structurant. L'élément structurant est donné par son *num_se* et sa demi-taille par *halfsize*.

La dilatation correspond à l'opération:

```
dilatation(x,y)= Max(voisins selon l'élément structurant de x,y).
```

Pour une image binaire cela revient à dilater les régions blanches. Pour les cartes de régions, la dilatation des régions ne s'effectue que sur le fond en privilégiant les régions de plus fort label.

Pour les images couleur, c'est l'ordre lexicographique qui est utilisé: d'abord en utilisant la bande X, en cas d'égalité en utilisant la bande Y puis la bande Z.

Paramètres

- *num_se* spécifie le type de l'élément structurant :

  En 2D:

  - 0: losange (4-connexité)
  - 1: carré (8-connexité).
  - 2: cercle
  - 3: ligne horizontal
  - 4: ligne diagonale de 135 degrés (\)
  - 5: ligne verticale
  - 6: ligne diagonale de 45 degrés (/)
  - 7: croix (+)
  - 8: X

  En 3D:

  - 0: bipyramide (6-connexité)
  - 1: cube (26-connexité)
  - 2: sphère
  - 3: ligne horizontale sur l'axe x

- 4: ligne horizontale sur l'axe y
- 5: ligne horizontale sur l'axe z
- 6: ligne diagonale sur le repère x-y (\)
- 7: ligne diagonale sur le repère x-z (\)
- 8: ligne diagonale sur le repère y-z (\)
- 9: ligne diagonale sur le repère x-y (/)
- 10: ligne diagonale sur le repère x-z (/)
- 11: ligne diagonale sur le repère y-z (/)
- 12: croix en 3d

(Ce paramètre est ignoré pour les images 1D)

- *halfsize* donne la demi-taille de l'élément structurant. Par exemple, une demi-taille de 1 pour un carré donne un structurant de taille 3x3.

Entrées

- *im_in*: an image (1D, 2D, 3D) or a region map.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Chapeau haut-de-forme noir avec un élément structurant carré de taille 17.

```
pinverse tangram.pan i0.pan
pdilation 1 8 i0.pan i1.pan
perosion 1 8 i1.pan i2.pan
pdif i2.pan i0.pan out.pan
```

Voir aussi

Morphologie, psedilation, perosion

Prototype C++

```
Errc PDilation( const Img2duc &im_in, Img2duc &im_out, int num_se,
int halfsize );
```

Auteur: Régis Clouard

pdilationreconstruction

Reconstruction par dilatation.

Synopsis

pdilationreconstruction *connexity* [*-m mask*] [*im_in1*|-] [*im_in2*|-][*im_out*|-]

Description

L'opérateur **pdilationreconstruction** effectue une reconstruction géodésique par dilatation de l'image de marqueurs *im_in1* dans l'image masque *im_in2*.

Les deux images doivent être de même type, mais l'image de marqueurs *im_in1* doit être inférieure ou égale en intensité à l'image de masque *im_in2*.

La reconstruction par dilatation selon la connexité *connexity* consiste en l'opération appliquée jusqu'à idempotence :

```
im1=min(im_in1,im_in2)
imerod=dilation(im1,connexity)
im_out=min(imerod,im_in2)
```

Pour les images couleur, c'est l'ordre lexicographique qui est utilisé : d'abord en utilisant la bande X, en cas d'égalité en utilisant la bande Y puis la bande Z.

Paramètres

- *connexity* définit la connexité par la relation de voisinage entre pixels : 2 pour le 1D, 4 ou 8 voisinage pour le 2D et 6 ou 26 voisinage pour le 3D.

Entrées

- *im_in1* : une image.
- *im_in2* : une image de même type que l'image d'entrée *im_in1*.

Sorties

- *im_out* : un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Chapeau haut-de-forme blanc géodésique avec un élément structurant carré taille 17.

```
perosion 1 8 tangram.pan i1.pan
pdilationreconstruction 8 i1.pan tangram.pan i2.pan
pdif tangram.pan i2.pan out.pan
```

Voir aussi

Morphologie, perosionreconstruction.

Prototype C++

```
Errc PDilationReconstruction( const Img2duc &im_in1, const Img2duc
&im_in2, Img2duc &im_out, int connexity );
```

Auteur : Régis Clouard

pdisplayperformancevalues

Affichage détaillé des erreurs de segmentation calculées par l'opérateur 'passesssegmentationalgorithm'.

Synopsis

pdisplayperformancesvalues [*col_in1*|-] [*col_in2*|-]

Description

L'opérateur **pdisplayperformancevalues** dresse un bilan détaillé des performances d'un algorithme calculées par l'opérateur *passesssegmentationalgorithm*. Les performances sont décrites par les erreurs de segmentation selon cinq indicateurs :

- **Indicateur 1** : La **précision de la détection** : Les deux erreurs sont :
  - L'erreur de rappel qui rend compte de la proportion de faux négatifs.
  - L'erreur de précision qui rend compte de la proportion de faux positifs.
- **Indicateur 2** : La **cohérence de la fragmentation** : Les deux erreurs sont :
  - L'erreur de sous-segmentation qui rend compte de la proportion de régions agglomérées par segment.
  - L'erreur de sur-segmentation qui rend compte de la proportion de fragmentation des régions en plusieurs segments.
- **Indicateur 3** : La **localisation des frontières** : Les deux erreurs sont :
  - L'erreur de déficit de pixels qui rend compte de la proportion de pixels non détectés dans les régions détectées.
  - L'erreur d'excès de pixels qui rend compte de la proportion de pixels erronés ajoutés aux régions détectées.
- **Indicateur 4** : Le **respect de la forme** : Les deux erreurs sont :
  - L'erreur de forme due à l'omission de surface des régions.
  - L'erreur de forme due à l'ajout de surface aux régions.
- **Indicateur 5** : La **préservation de la topologie** : Les deux erreurs sont :
  - L'erreur d'ajout de trou qui rend compte de la proportion de faux trous détectés.
  - L'erreur de suppression de trou qui rend compte de la proportion de trous non détectés.

Entrées

- *col_in1*: une collection avec le détail des valeurs d'erreur de segmentation pour chaque résultat de segmentation (stocké sous la forme numérateur, dénominateur).
- *col_in2*: une collection avec les valeurs moyennes d'erreur de segmentation en considérant tous les résultats de segmentation.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Évaluation de la qualité de l'algorithme 'algo001' à partir de ses résultats stockés dans le dossier 'images/resulimages/algo001:

```
passesssegmentationalgorithm 0 0.5 images/resultimages/algo001 images/groundtruths detail_errors.pan total_errors.
pdisplayperformancevalues detail_errors.pan total_errors.pan
```

Voir aussi

passesssegmentationalgorithm, Evaluation

Prototype C++

```
Errc DisplayPerformanceValues( const Collection &cols1, const
Collection &cols2 );
```

Auteur: Régis Clouard

pdistance

Calcul d’une image de distance euclidienne aux contours.

Synopsis

pdistance [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L’opérateur **pdistance** calcule pour chaque point de l’image *im_in*, sa distance au contour le plus proche. Les contours de l’image *im_in* sont des chaînes de pixels de valeur non nulle reposant sur un fond de valeur nulle.

L’image *im_out* est une image réelle, où chaque pixel indique la valeur entière de la distance de ce point au contour le plus proche.

En 2D, l’algorithme utilisé repose sur la distance euclidienne exacte proposée par Meijster.

En 3D, l’algorithme utilisé repose sur la résolution de l’équation Eikonale (fast marching). Elle correspond alors à une bonne approximation de la distance euclidienne.

Reference: A. Meijster, J. B. T. M. Roerdink and W. H. Hesselink, "A general algorithm for computing distance transforms in linear time.", In: Mathematical Morphology and its Applications to Image and Signal Processing, Kluwer Acad. Publ., 2000, pp. 331-340.

Entrées

- *im_in*: une image de niveaux de gris ou un graphe.

Sorties

- *im_out*: une image réelle ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Ferme les contours obtenus par une simple détection de contours :

```
psobel tangram.pan b.pan
pbinarization 50 1e30 b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
pdistance e.pan f.pan
plocalmaxima 8 f.pan g.pan
plabeling 8 g.pan h.pan
pinverse f.pan i.pan
pwatershed h.pan i.pan j.pan
pboundary 8 j.pan out.pan
```

Voir aussi

Contour

Prototype C++

```
Errc PDistance( const Img2duc &im_in, Img2dsf &im_out );
```

Auteurs: Jean-Marie Janik & Abderrahim Elmoataz

pdistance1

Calcul d'une image de distance quelconque aux contours.

Synopsis

pdistance1 *d1 d2 d3* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pdistance1** calcule pour chaque point de l'image *im_in*, sa distance au contour le plus proche.

Les contours de l'image *im_in* sont des chaînes de pixels de valeur non nulle reposant sur un fond de valeur nulle.

La distance est calculée en 8 voisinage pour le 2D selon les valeurs *d1* et *d2* et en 26 voisinage pour le 3D selon les valeurs *d1*, *d2* et *d3* données en paramètres.
Le principe de calcul est le suivant:

- Si le pixel est un point de contour alors distance = 0.
- Sinon, la distance est le minimum de la distance de ces voisins + d1 s'il est dans son 4 voisinage (voisin 0, 2 , 6) ou + d2 sinon, comme ci-dessous:

```
+d2 +d1 +d2
+d1  x  +d1
+d2 +d1 +d2
```

L'image de sortie *im_out* est une image réelle, où chaque pixel indique la valeur entière de la distance de ce point au contour le plus proche.

Paramètres

- *d1*, *d2* et *d3* sont des réels qui permettent de spécifier la métrique. En 2D, *d3* n'est pas utilisé, mais doit être donné.
  Les métriques plus utilisées sont:
  - Euclidienne : d1 = 1 ; d2 = sqrt(2), d3 = sqrt(3);
  - Jeux d'echec : d1 = 1 ; d2 = 2; d3 = 3;
  - Manhattan : d1 = 1 ; d2 = 1; d3 = 1;

Entrées

- *im_in*: une image de niveaux de gris ou un graphe.

Sorties

- *im_out*: une image réelle ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Ferme les contours obtenus par une simple détection de contours :

```
psobel tangram.pan b.pan
pbinarization 50 1e30 b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
pdistance1 1 1 1 e.pan f.pan
plocalmaxima 8 f.pan g.pan
plabeling 8 g.pan h.pan
pinverse f.pan i.pan
pwatershed h.pan i.pan j.pan
pboundary 8 j.pan out.pan
```

Voir aussi

Contour

Prototype C++

```
Errc PDistance1( const Img2duc &im_in, Img2dsf &im_out, float d1,
float d2, float d3 );
```

Auteur: Régis Clouard

pdiv

Division d'images ou de graphes.

Synopsis

pdiv [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pdiv** effectue la division de l'image *im_in1* par l'image *im_in2*. L'opération est faite entre chaque pixel de l'image *im_in1* et de l'image *im_in2* correspondant.

Le problème du zéro au dénominateur est géré en utilisant un epsilon. Il n'y a pas de gestion du débordement de valeurs. La formule reprend exactement l'opérateur division du C :

```
if (pixel(im_in) == 0 )
   pixel(im_out)= 0;
else
   pixel(im_out) = pixel(im_in1)  / pixel(im_in2);
```

Les deux images d'entrée *im_in1* ou *im_in2* doivent être de même type. Au besoin, il est nécessaire d'utiliser les opérateurs de coercition. Par contre, l'image de sortie est de type réel (Float).

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Entrées

- *im_in1*: une image ou un graphe.
- *im_in2*: une image ou un graphe.

Sorties

- *im_out*: une image de Floats ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Divise l'image a.pan par l'image b.pan :

```
pdiv a.pan b.pan c.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PDiv( const Img2duc &im_in1, const Img2duc &im_in2, Img2dsf
&im_out );
```

Auteur: Régis Clouard

pdivcst

Division par une constante des valeurs d'une image, d'un graphe ou d'une carte de région.

Synopsis

pdivcst *cst* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pdivcst** calcule l'image *im_out* par division des valeurs de pixels de l'image *im_in* par la valeur *cst*.

Dans tous les cas, l'image de sortie est du même type que l'image d'entrée.

Il y a écrétage du résultat si la valeur résultante est supérieure à la valeur maximale du type de l'image.

La formule de calcul est la suivante :

```
pixel(im_out) = pixel(im_in) / cst;
```

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Pour une carte de régions, ce sont les valeurs des labels qui sont divisées.

Pour un graphe, ce sont les valeurs des noeuds qui sont divisées.

Paramètres

- *cst* est une valeur réelle.

Entrées

- *im_in*: une image, un graphe ou une carte de régions

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Pour les cartes de régions, retourne la valeur de label maximale.

Exemples

Divise les pixels de l'image tangram.pan par 2:

```
pdivcst 2 tangram.pan a.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PDivCst( const Img2duc &im_in, Img2duc &im_out, float cst );
```

Auteur: Régis Clouard

pdivneumann

Calcul de la divergence par différences finies décentrées à gauche.

Synopsis

pdivneumann [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pdivneumann** calcule la divergence par différence finie décentrée à gauche. Le résultat est une image de niveaux de gris *im_out*, où:

```
 im_out(i,j) = (im_in1(i,j)-im_in1(i-1,j)) + (im_in2(i,j)-im2(i,j-1)),
```

avec les conditions de Neumann aux bords:

```
im_in1(1,j) et -im_in1(n-1,j)
im_in2(i,1) et -im_in2(i,n-1)
```

Entrées

- *im_in1*: une image 2D.
- *im_in2*: une image 2D (de mêe type que im_in1).

Sorties

- *im_out*: un image de même type que im_in1.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Implémente le calcul du gradient et de la divergence avec les condition aux bords de Neumann, telle que façon que l'une est l'adjoint de l'autre, i.e. <grad x,u> = <-div u,x>. Le script vérifie l'identité de deux images.

```
protation 0 180 tangram.pan tangram1.pan
pdivneumann tangram.pan gim1_y.pan gim1_x.pan
pdivneumann tangram1.pan gim2_y.pan gim2_x.pan

# Compute < grad im1, grad im2>.
pmult gim1_y.pan gim2_y.pan | psumvalue - s1.pan
sumvaly=‘pstatus‘
pmult gim1_x.pan gim2_x.pan | psumvalue - s2.pan
```

```
sumvalx=‘pstatus‘

innerproduct1=‘echo "$sumvaly+$sumvalx" | bc -l‘

# Compute <-div grad im1,im2>.
pdivneumann gim1_y.pan gim1_x.pan | pmultcst -1 - divim1.pan
pim2sf tangram1.pan t.pan
pmult divim1.pan t.pan | psumvalue - /dev/null
innerproduct2=‘pstatus‘

echo $innerproduct1
echo $innerproduct2
```

Voir aussi

Edge detection, pdivneumann

Prototype C++

```
Errc PDivNeumann( const Img2d<U> &im_in1, Img2d<U> &im_in2, Img2d<U> &im_out );
```

Auteur: Jalal Fadili

pdivval

Division d'une image par des constantes stockées dans une collection.

Synopsis

pdivval [*-m mask*] [*col_in*|-] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pdivval** calcule l'image *im_out* par division des valeurs de pixels de l'image *im_in* par les valeur stockes dans la collection *col_in*. La première valeur de la collection est utilisée pour diviser tous les pixels de la première bande, la seconde à tous les pixels de la seconde bande, etc.

Il y a écrétage du résultat si la valeur résultante est supérieure à la valeur maximale du type de l'image. La formule de calcul est la suivante :

```
val = pixel(im_in) / col_in;
if (val > MAX) pixel(im_out) = MAX;
else if (val < MIN) pixel(im_out) = MIN;
else pixel(im_out) = val;
```

Entrées

- *col_in*: une collection avec autant de valeurs réelles que de nombre de bandes pour l'image d'entrée (p. ex. 3 pour une image couleur).
- *im_in*: une image.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Divise tangram.pan sa valeur moyenne:

```
pmeanvalue tangram.pan a.pan
pdivval a.pan tangram.pan b.pan
```

Autres exemples

Voir aussi

Arithmetique

Prototype C++

```
Errc PDivVal( const Collection &col_in, const Img2duc &im_in,
Img2duc &im_out );
```

Auteur: Régis Clouard

pdraw

Interface de dessin sur une image.

Synopsis

pdraw [-m mask] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pdraw** permet de dessiner des contours sur un objet Pandore. Les objets Pandore utilisables sont :

- images;
- regions map;
- graphs.

La version Qt de **pdraw** accepte aussi les options QT au début de la liste des arguments:

```
pdraw -style motif tangram.pan a.pan
```

Entrées

- *im_in*: une image, une carte de régions ou un graphe.

Entrées

- *im_out*: une image 2D ou 3D d'octets (Img2duc).

Résultat

Retourne le numéro du processus (PID) ou FAILURE.

Exemples

- Crée l'image a.pan à partir d'un dessin sur l'image tangram.pan puis l'étiquetage des régions obtenus :

```
pdraw examples/tangram.pan a.pan
pboundarylabeling a.pan b.pan
pvisu b.pan
```

Voir aussi

Visualisation, pvisu.

Auteur: Régis Clouard

pdwt

Calcul de la transformée en ondelettes dyadiques biorthogonales d'une image.

Synopsis

pdwt *scale* [*im_in*|-] [*col_in*|-] [*im_out*| -]

Description

L'opérateur **pdwt** calcule les coefficients d'ondelette d'une image en niveaux de gris selon l'algorithme pyramidal. Par exemple, à l'*echelle* 1, on obtient alors 4 sous-images:

```
[1][2]
[3][4]
```

où [1] correspond à l'image approximée sous-échantillonée d'un facteur 2, et [2], [3], [4] correspondent au signal de détail suivant chacun une direction privilégiée (resp. horizontale, verticale, diagonale) sous-échantillonée d'un facteur 2.

Les coefficients du filtre utilisé se trouvent dans la collection*im_in* créée à partir de l'opérateur pqmf.

Important: L'opérateur fonctionne quelles que soient les dimensions de l'image. Néanmoins, l'algorithme exige d'avoir des images dont les dimensions sont en puissance de 2 (ex: 128, 256, 512,...). De ce fait, l'opérateur utilise l'image initiale en ajoutant des bandes noires sur la droite et en dessous de l'image initiale pour que les dimensions soient en puissance de 2. Ainsi, si l'une des dimensions est comprise entre 129 et 256, elle sera de 256 dans l'image finale, etc...

Paramètres

- *scale* indique le nombre de niveaux de résolution sur lesquels on veut décomposer l'image.

Entrées

- *im_in*: une image 2D.
- *col_in*: une collection qui contient les coefficients du filtre.

Sorties

- *im_out*: une image 2D de Float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit une image synthétique avec un carré pour illustrer le phénomène de Gibbs an analyse par ondelettes:

```
pshapedesign 256 256 0 2 150 150 a.pan
pqmf daubechies 4 b.pan
pdwt 1 a.pan b.pan c.pan
psplitimage c.pan d1.pan d2.pan d3.pan d4.pan
pthresholding 20 400 d2.pan e2.pan
pthresholding 20 400 d3.pan e3.pan
pthresholding 20 400 d4.pan e4.pan
pmergeimages d1.pan e2.pan e3.pan e4.pan f.pan
pidwt 1 f.pan b.pan out.pan
```

Voir aussi

Domaine Fréquentiel, pidwt, pqmf

Prototype C++

```
Errc PDwt( const Img2duc &im_in, const Collection &col_in, Img2dsf
&im_out, int scale );
```

Auteur: Ludovic Soltys

peccentricityselection

Sélection de régions sur leur valeur d'excentricité.

Synopsis

peccentricityselection *relation seuil* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **peccentricityselection** permet de sélectionner les régions sur leur degré d'excentricité. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

L'excentricité est calculée par:

$$\text{excentricite} = \frac{(Mxx+Myy-\sqrt{(Mxx-Myy)*(Mxx-Myy)+4*Mxy*Mxy})}{(Mxx+Myy+\sqrt{(Mxx-Myy)*(Mxx-Myy)+4*Mxy*Mxy})}$$

Elle correspond au rapport de la longueur du petit axe sur celle du grand axe d'une région. Le résultat est une valeur réelle entre 0 et 1, avec 1 pour un carré.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil*est une valeur réelle de l'intervalle [0,1].

Entrées

- *rg_in*: une carte de régions 2D

Sorties

- *rg_out*: une carte de régions 2D

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Exemples

Sélectionne les régions de plus forte excentricité :

```
peccentricity 3 0 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PEccentricitySelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, Ushort seuil );
```

Auteur: Régis Clouard

pedgebasedragpruning

Séparation dans le graphe d'adjacence de régions séparées par un point de contour.

Synopsis

pedgebasedragpruning [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*gr_out*|-]

Description

L'opérateur **pedgebasedragpruning** permet de couper le lien d'adjacence du graphe *gr_in* entre deux régions qui sont séparés par un point de contours donné dans l'image *im_in* (un pixel de valeur >0).

Cet opérator permet de guider un processus de fusion en empéchant de fusionner deux régions séparées par un contour.

Entrées

- *rg_in*: une carte de regions.
- *gr_in*: un graphe.
- *im_in*: une carte de contours.

Sortie

- *gr_out*: un graphe.

Résultat

Retourne le nombre cuts.

Exemples

Processus ascendant de fusion de régions:

```
pbinarization 112 255 tangram.pan a.pan
plabeling 8 a.pan b.pan
paddcst 1 b.pan c.pan
prg2gr c.pan d.pan

# sans les contours.
psetcst 0 tangram.pan e.pan
pmeanmerging -1 60 c.pan d.pan e.pan f1.pan g1.pan

# avec les contours
```

```
pderiche 1 tangram.pan f.pan g.pan
pvariancebinarization f.pan h.pan
pedgebasedragpruning c.pan d.pan h.pan i.pan
pmeanmerging -1 60 c.pan i.pan e.pan f2.pan g2.pan
```

Voir Aussi

Segmentation

Prototype C++

```
Errc PEdgeBasedRAGPruning( const Reg3d &rg_in, const Graph3d &gr_in,
const Img3duc &im_in, Graph3d &gr_out);
```

Auteur: Régis Clouard

pedgeclosing

Fermeture de contours par poursuite du gradient.

Synopsis

pedgeclosing *angle longueur* [*-m mask*] [*im_in*|-] [*im_amp*|-] [*im_out*|-]

Description

L'opérateur **pedgeclosing** consiste à fermer les contours donnés dans l'image *im_in*, par poursuite selon le gradient maximum donné dans l'image *im_amp*.

A partir des points terminaux, la poursuite se fait en prenant la direction du gradient maximum dans les directions de poursuite limitées par le paramètre *angle* qui spécifie l'écart maximum autorisé avec la normale de la poursuite.

Si le gradient est nul alors la poursuite s'arrête.
L'image de sortie est fermée par un contour sur le bord.

Cet opérateur nécessite que les points terminaux n'aient qu'un seul voisin. Il peut donc être utile d'utiliser l'opérateur ppostthinning qui amincit les contours en ne gardant que la 8-connexité.

Paramètres

- Le paramètre *angle* permet de spécifier l'angle de recherche du point suivant. Il appartient à l'intervalle [0..2].
  - Si *angle*=0 alors la poursuite se fait dans la même direction que la fin du contour (0 degré de liberté).
  - *angle*=1 correspond à 0, 45 et -45 degrés.
  - *angle*=2 correspond à 0, 45, 90, -45, -90 degré.
- La *longueur* détermine la longueur maximale autorisée pour la poursuite.

Entrées

- *im_in*: une image 2D de type Uchar.
- *im_amp*: une image entière 2D.

Sorties

- *im_out*: une image 2D de Uchar.

Résultat

Retourne le nombre de contours fermés ou FAILURE.

Exemples

Ferme les contours obtenus par une simple detection de contours :

```
psobel tangram.pan b.pan
pbinarization 50 1e30 b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
pedgeclosing 1 10 e.pan b.pan out.pan
pstatus
```

Voir aussi

Contour

Prototype C++

```
Errc PEdgeClosing( const Img2duc &im_in, const Img2duc &ima, Img2duc &im_out, int angle, int longueur)
```

Auteur: Régis Clouard

pedgecutting

Suppression des arêtes d'un graphe sur la valeur.

Synopsis

pedgecutting *low high* [*-m mask*] [*gr_in*|-] [*gr_out*|-]

Description

L'opérateur **pedgecutting** procède à la suppression physique des arcs entre deux sommets dont la valeur est comprise entre la valeur de seuil bas *low* et de seuil haut *high*.

Si *high* est inférieur à *low* alors la coupure est faite pour les valeurs inférieures au seuil haut *high*> et supérieures au seuil bas *low*.

Paramètres

low et *high* permettent de spécifier la zone de valeur de coupe.

Si *high* est inférieur à *low* alors la coupure est faite pour les valeurs inférieures au seuil haut *high*> et supérieures au seuil bas *low*.

Entrées

- *gr_in*: un graphe.

Sorties

- *gr_out*: un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Conserve les arêtes avec une valeur comprise entre 1 et 2 :

```
pedgecutting 2 1 g1.pan g2.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PEdgeCutting( const Graph &gr_in, Graph &gr_out, float low,
float high );
```

Auteur: François Angot

pedgedirection

Calcul de la direction des contours.

Synopsis

pedgedirection [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pedgedirection** calcule la direction des contours. Un contour est une ligne formée de pixels à 255, et le fond est à 0.

L'algorithme est basée sur une approximation de la normale aux contours.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image du même type que l'image im_in.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Détection de contours basée sur le seuillage par hystérésis puis calcul des directions de contours (peut être comparée avec l'image i2.pan):

```
pgradient 1 tangram.pan i1.pan i2.pan
pbinarization 30 1e30 i1.pan i3.pan
pbinarization 60 1e30 i1.pan i4.pan
pedgedirection i4.pan out.pan
```

Voir aussi

Détection de contours

Prototype C++

```
Errc PEdgeDirection( const Img2duc &im_in, Img2duc &im_out);
```

Auteur: Régis Clouard

pedgevisu

Visualisation des poids des arêtes d'un graphe dans une image.

Synopsis

pedgevisu [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_out*|-]

Description

L'opérateur **pedgevisu** permet de visualiser les valeurs des arêtes dans un graphe.

Chaque frontière de la carte *rg_in* est tracée en utilisant la couleur correspondant au poids attribué à l'arête entre les deux régions.

Entrées

- *gr_in*: un graphe.

Sorties

- *gr_out*: une image.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pbinarization 90 1e30 tangram.pan a.pan
plabeling 8 a.pan r1.pan
pcenterofmass r1.pan r2.pan
pdelaunay r2.pan g2.pan
pedgevisu r2.pan g2.pan out.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PEdgeVisu( const Reg2d &rg_in, const Graph2d &gr_in, Img2dsl
&im_out );
```

Auteur: François Angot

pelliptisoidalapproximation

Approximation ellipsoïdale d'un ensemble de points ou des contours d'une image.

Synopsis

`pelliptisoidalapproximation` *`mode`* `m mask`] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pelliptisoidalapproximation** consiste à approximer un ensemble de points par une ellipse. Les points sont repérés par une valeur non nulle (valeur entre [1..255]) reposant sur un fond nul (valeur =0).

L'image de sortie *im_out* est au format Uchar (octet) où les points de contours de l'ellipse sont à 255.

L'elliptisation opère sur tous les points si *mode*=0 ou sur chacune des chaînes de contours si *mode*=1. Une chaîne de contours est une séquence 8-connexe de pixels non nuls.

En sortie, les ellipses sont marquées par des pixels à 255 reposant sur un fond nul.

Paramètres

- Le *mode* est un entier entre 0 et 1 qui précise:
  - mode=0: l'ellipse doit être calculée sur tous les points de l'image;
  - mode=1: l'ellipse doit être calculée sur chacune des chaînes de contours.

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: une image de Uchar.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Approxime chaque pièce de tangram par une ellipsoïde :

```
psobel tangram.pan b.pan
pbinarization 45 1e30 b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
pclosedcontourselection 1 50 e.pan f.pan
pellipsoidalapproximation 1 f.pan out.pan
```

Voir aussi

Contour

Prototype C++

```
Errc PEllipsoidalApproximation( const Img2duc &im_in, Img2duc
&im_out, int mode );
```

Auteur: Julien Robiaille

pelongationselection

Sélection de régions sur leur valeur d'élongation.

Synopsis

pelongationselection *relation seuil* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pelongationselection** permet de sélectionner les régions sur leur degré d'élongation. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

L'élongation est le rapport entre la largeur et la longueur du rectange exincrit des régions (la longueur est le plus grand côté et la largeur est le plus petit:

```
elongation = largeur(rectangle)/longueur(rectangle).
```

Le rectangle exinscrit est calculé avec différentes orientations, et on conserve celui qui donne l'élongation minimale, en considérant que c'est celui qui épouse le mieux la forme.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur réelle [0..1] qui correspond au facteur d'élongation.
  Cette valeur est égale à 1 pour un carré ou un disque, et <<1 pour un objet oblongue.

Entrées

- *rg_in*: une carte de régions

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de régions ainsi sélectionnée.

Exemples

Sélectionne les régions les plus allongées :

```
pelongationselection 3 0 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PElongationSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, float seuil );
```

Auteur: Régis Clouard

penergyselection

Sélection de régions sur leur valeur d'énergie intérieure.

Synopsis

penergyselection *relation seuil* [*-m mask*] [*rg_in*|-] [*im_in*|-][*rg_out*|-]

Description

L'opérateur **penergyselection** permet de sélectionner les régions sur leur énergie intérieure calculée dans *im_in*. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

La mesure d'énergie est faite selon la formule:

```
energie = SOMME{ im_in[p] * im_in[p]} / N
```

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.

penergyvalue

Calcul de l'énergie d'une image ou d'un graphe.

Synopsis

penergyvalue [*im_in*|-] [*col_out*|-]

Description

L'opérateur **penergyvalue** mesure l'énergie de l'image d'entrée *im_in*.

La mesure d'énergie est faite selon la formule:

```
energie = SOMME{ im_in[p] * im_in[p]} / N
```

Pour une image, l'énergie est mesurée par bande à partir des valeurs de pixel.

Pour les graphes, l'énergie est mesurée à partir des valeurs des noeuds.

Les valeurs d'énergie de chaque bande sont stockées dans la collection *col_out*.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *col_out*: une collection de valeurs réelles.

Résultat

Retourne la valeur d'énergie globale (pour la première bande uniquement).
La valeur est accessible par la commande **pstatus**.

Exemples

Mesure l'énergie globale de l'image tangram.pan (version Unix):

```
penergyvalue tangram.pan col.pan
val=‘pstatus‘
echo "Energie = $val"
```

Mesure l'énergie globale de l'image tangram.pan (version MsDos):

```
penergyvalue tangram.pan col.pan
call pstatus
call pset val
echo Energie = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Float PEnergyValue( const Img2duc &im_in, Collection & col_out );
```

Auteur: Régis Clouard

pentropybinarization

Binarisation de l'image par maximisation de l'entropie interclasse.

Synopsis

pentropybinarization [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pentropybinarization** classifie les pixels de l'image d'entrée *im_in* en 2 deux classes. La valeur de seuil est déterminée comme la valeur de niveau de gris `s` qui maximise la quantité totale d'information du fond et des objets pris séparément. Puisque l'information se mesure par l'entropie, la quantité totale d'information pour un seuil s est donnée par :

```
TE(s) = Eb(s) + Ef(s) { entropie du fond + entropie des objets }
      = ln[P(s)(1-P(s))] - H(s)/P(s) - H'(s)/(1-P(s))
  où P(s) = SUM{i=0->s} [p(i)]
  et H(s) = SUM{i=0->s} [p(i)*ln(pi)]
  et H'(s) = SUM{i=s->m-1} [(p(i)*ln(pi)]
  et W*H estle nombre de pixels
  et m est le nombre de niveaux de gris.
  et pi = fi/W*H
```

Le critère d'entropie maximale se détermine à partir du seuil smax tel que:

```
TE(smax) = max TE(s)
```

Entrées

- *im_in*: une image niveaux de gris de Uchar (Img2duc, Img3duc).

Sorties

- *im_out*: une image niveaux de gris de Uchar (Img2duc, Img3duc).

Résultat

Retourne la valeur de seuil détectée.

Exemples

Segmentation des pièces de tangram :

```
pentropybinarization tangram.pan a.pan
```

Voir aussi

Thresholding

Prototype C++

```
Errc PEntropyBinarization( const Img2duc &im_in, Img2duc &im_out );
```

Référence

J-C Yen, F-J Chang, S. Chang, "A New Criterion for Automatic Multilevel Thresholding", *IEEE Trans. on Image Processing*, vol. 4, no. 3, pp 370-378, 1995.

Auteur: Régis Clouard

pentropymerging

Fusion prioritaire de régions selon le critère de l'entropie.

Synopsis

pentropymerging *nb_fusion seuil* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

L'opérateur **pentropymerging** permet de fusionner les régions de la carte de régions *rg_in* selon le critère de l'entropie.

La notion de voisinage entre les régions est détenue par le graphe *gr_in*.

Le principe de l'algorithme est le suivant:
Pour chaque région de la carte de régions *im_in*, on calcule les différences entre la somme des entropies de la région et de ses voisines avec l'entropie de la somme de ces régions.

```
différence(R1,R2)= (entropie(R1)+entropie(R2)) - entropie(R1+R2)
```

Si une différence est supérieure au *seuil* donné en paramètre, alors les régions sont fusionnées.
On utilise ici l'algorithme de croissance prioritaire qui consiste à fusionner à chaque fois les 2 régions dont la différence est la plus faible.

Le calcul de l'entropie se fait selon la formule de Shannon:

```
entropie(R)=sum(Pi*log(Pi)) / log(2)
où Pi la probabilité du pixel sur la region soit
Pi = H[i]/ N, avec N le nombre de pixels de la region.
```

Paramètres

- *nb_fusion* permet de spécifier le nombre de fusion à effectuer (la valeur -1 signifie d'ignorer ce paramètre et d'exécuter l'algorithme tant qu'il y a des fusions possibles).
- *seuil* permet de spécifier la tolérance par rapport à la différence entre deux régions. Une valeur typique est 0.

Entrées

- *rg_in*: une carte de régions.
- *gr_in*: un graphe.
- *im_in*: une image.

Sorties

- *rg_out*: une carte de régions.
- *gr_out*: un graphe.

Résultat

Retourne le nombre de fusions effectuées.

Exemples

Fusionne les régions issues d'une partition de tangram.pan :

```
puniformityquadtree 0.9 tangram.pan a.pan
prg2gr a.pan b.pan
pentropymerging -1 -2 a.pan b.pan tangram.pan c.pan d.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PEntropyMerging( const Reg2d &rg_in, const Graph2d &gr_in,
const Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, Long
nb_fusion, Uchar seuil );
```

Auteur: Laurent Quesnel

pentropyquadtree

Segmentation d’une image par quadtree (octree) selon l’entropie.

Synopsis

pentropyquadtree *seuil* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

L’opérateur **pentropyquadtree** permet de segmenter l’image en différentes régions selon l’entropie. Les régions obtenues sont rectangulaires.

Le principe de l’algorithme est le suivant:

- Si un bloc n’est pas homogéne (i.e. l’entropie est supérieure au seuil) alors on divise le bloc en 4 blocs égaux et on réapplique l’algorithme sur chacun des blocs.

L’entropie d’une région mesure la quantité d’information portée par cette région. Plus un pixel est rare, plus il est porteur d’information et plus l’entropie est grande.

On utilise ici la valeur de l’entropie calculée par:

```
entropie(R)= -SOMME(p[i]*log(p[i])) / log(2),
avec p[i] probabilité d’avoir la valeur i dans l’image,
soit le (nombre de pixel=i) / N
où N est le nombre de pixels de l’image.
```

Cet opérateur ne peut être appliqué sur des images de float cat il faut que les probabilités associées aux pixels de cette image soit à peu près significatives, c’est à dire que les valeurs ne soient pas toutes différentes.

En 3D, le résultat est un octree, c’est à dire une carte de régions composée de cubes.

Paramètres

- *seuil* est la valeur d’entropie maximum pour qu’une région soit acceptée comme uniforme. Une valeur de 0 correspond à rechercher des régions totalement homogènes en entropie. La valeur maximale peut être approximée par ln(N)/Ln(2), où N est le nombre de pixels de l’image.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de régions obtenues.

Exemples

Construit une partition de tangram.pan:

```
pentropyquadtree 4 tangram.pan a.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PEntropyQuadtree( const Img2duc &im_in, Reg2d &rg_out, float
seuil );
```

Auteur: Laurent Quesnel

pentropythresholding

Multiseuillage de l'image par analyse de l'entropie des régions.

Synopsis

pentropythresholding *length* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pentropythresholding** permet de multiseuiller l'image initiale *im_in* par classification des pixels basée sur l'analyse de la matrice de co-occurences.

La matrice de co-occurence Tkl utilisée est telle que Tkl représente le nombre de fois où dans une fenêtre 3x3, le pixel central est au niveau k et la moyenne sur ses 8 voisins est égale à l.

Pour chaque niveau de gris k, la fonction d'entropie est:

```
Entropie(k) = - Somme(Tkl * Log (Tkl)) avec l=[0..k].
```

La recherche des maxima locaux de Entropie(k) sur une plage de *length* niveaux de gris de part et d'autre du niveau de gris k.

Remarque: Cet opérateur ne fonctionne que sur des images de Char parce qu'il faut que les transitions Tkl soient significatives (ie, nombre de (k,l) restreints).

L'image de sortie *im_out* est contruite avec les seuils détectés, telle que:

```
im_out[y][x]=seuil[k] si seuil[k-1]<im_out[y][x]<=seuil[k].
```

Le dernier seuil est égal à la valeur maximale 255.

Paramètres

- *length* définit la plage de recherche du minima de la fonction énergie. Plus ce nombre est élevé moins il y a de maxima régionaux et donc moins de classes en sortie. Une valeur typique est 10.

Entrées

- *im_in*: une image de niveaux de gris en octets (Img2duc, Im3duc).

Sorties

- *im_out*: une image de niveaux de gris en octets (Img2duc, Im3duc).

Résultat

Retourne le nombre de seuils détectés.

Exemples

Segmente l'image tangram.pan et affiche le nombre de classes détectées:

```
pentropythresholding 10 tangram.pan out.pan
pstatus
```

Voir aussi

Seuillage

Prototype C++

```
Errc PEntropyThresholding( const Img2duc &im_in, Img2duc &im_out,
Ushort * seuils, int length );
```

éférence

C. Fernandez-Maloigne, "*Segmentation et caractérisation d'images de textures à l'aide d'informations statistiques*", PhD Thesis, University of Compiegne, 1989.

Auteur: Régis Clouard

pentropyvalue

Calcul de l'entropie d'une image ou d'un graphe.

Synopsis

pentropyvalue [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pentropyvalue** permet de calculer l'entropie totale d'une image ou d'une graphe.

L'entropie au sens de Shannon mesure la qualité d'information portée par l'image *im_in*. Plus un pixel est rare, plus il est porteur d'information et plus l'entropie est grande (i.e., le désordre est important).

La mesure d'entropie est faite selon la formule:

```
entropie= - SOMME { Pi * log2(Pi) }
où Pi est la probabilité d'apparition du niveau de gris i.
(Pi est calculé à partir de l'histogramme normalisé).
```

Les valeurs d'entropie de chaque bande sont stockées dans la collection *col_out*.

Les valeurs moyennes de chaque bande sont stockées dans la collection *col_out*.

Remarque: Cet opérateur ne fonctionne pas sur les images de réels parce que dans ce cas la probabilité d'apparition d'un pixel est pratiquement de 1/N. Il faut alors normaliser l'image avec d'autres opérateurs pour la convertir en image d'entiers.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *col_out*: une collection de valeurs réelles.

Résultat

Retourne la valeur d'entropie globale.
La valeur est accessible par la commande **pstatus**.

Exemples

Mesure l’entropie globale de l’image tangram.pan (version Unix):

```
pentropyvalue tangram.pan col.pan
var=‘pstatus‘
echo "Entropie = $val"
```

Mesure l’entropie globale de l’image tangram.pan (version MsDos):

```
pentropyvalue tangram.pan col.pan
call pstatus
call pset var
echo Entropie = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Errc PEntropyValue( const Img2duc &im_in, Collection & col_out );
```

Auteur: Régis Clouard

perosion

Erosion des points de fort contraste d'une image.

Synopsis

perosion *num_se halfsize* [*-m mask*][*im_in*|-][*im_out*|-]

Description

L'opérateur **perosion** permet d'éroder les points de plus fort contraste selon un élément structurant. L'élément structurant est donné par son numéro *num_se* et sa demi-taille par *halfsize*.

L'érosion correspond à l'opération:

```
erosion(x,y)= Min(voisins selon l'élément structurant de x,y).
```

Pour une image binaire cela revient à éroder les régions blanches.

Pour les cartes de régions, l'érosion ajoute des pixels de label=0 aux points d'érosion.

Pour les images couleur, c'est l'ordre lexicographique qui est utilisé: d'abord en utilisant la bande X, en cas d'égalité en utilisant la bande Y puis la bande Z.

Paramètres

- *num_se* spécifie le type de l'élément structurant :

  En 2D:

  - 0: losange (4-connexité)
  - 1: carré (8-connexité).
  - 2: cercle
  - 3: ligne horizontal
  - 4: ligne diagonale de 135 degrés (\)
  - 5: ligne verticale
  - 6: ligne diagonale de 45 degrés (/)
  - 7: croix (+)
  - 8: X

  En 3D:

  - 0: bipyramide (6-connexité)
  - 1: cube (26-connexité)
  - 2: sphère

    - 3: ligne horizontale sur l'axe x
    - 4: ligne horizontale sur l'axe y
    - 5: ligne horizontale sur l'axe z
    - 6: ligne diagonale sur le repère x-y (\)
    - 7: ligne diagonale sur le repère x-z (\)
    - 8: ligne diagonale sur le repère y-z (\)
    - 9: ligne diagonale sur le repère x-y (/)
    - 10: ligne diagonale sur le repère x-z (/)
    - 11: ligne diagonale sur le repère y-z (/)
    - 12: croix en 3d
    -

  (Ce paramètre est ignoré pour les images 1D)
- *halfsize* donne la demi-taille de l'élément structurant. Par exemple, une demi-taille de 1 pour un carré donne un élément structurant de taille 3x3.

Entrées

- *im_in*: une image (1D, 2D, 3D) ou une carte de régions.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Chapeau haut de forme blanc avec un élément structurant carré de taille 17.

```
perosion 1 8 tangram.pan i1.pan
pdilation  1 8 i1.pan i2.pan
pdif i2.pan tangram.pan out.pan
```

Voir aussi

Morphologie, pseerosion, pdilation

Prototype C++

```
Errc PErosion( const Img2duc &im_in, Img2duc &im_out, int num_es,
int halfsize );
```

Auteur: Régis Clouard

perosionreconstruction

Reconstruction morphologique par érosion.

Synopsis

perosionreconstruction *connexity* [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **perosionreconstruction** effectue une reconstruction géodésique par érosion de l'image de marqueurs *im_in1* dans l'image masque *im_in2*.

Les deux images doivent être de même type, mais l'image de marqueurs *im_in1* doit être supérieure ou égale en intensité à l'image de masque *im_in2*.

La reconstruction par érosion selon la connexité *connexity* consiste en l'opération appliquée jusqu'à idempotence :

```
im1=max(im_in1,im_in2)
imerod=erosion(im1,connexity)
im_out=max(imerod,im_in2)
```

Pour les images couleur, c'est l'ordre lexicographique qui est utilisé : d'abord en utilisant la bande X, en cas d'égalité en utilisant la bande Y puis la bande Z.

Paramètres

- *connexity* définit la connexité entre 2 pour le pour le 1D, 4 ou 8 voisinage pour le 2D et 6 et 26 voisinage pour le 3D.

Entrées

- *im_in1* : une image.
- *im_in2* : une image du même type que *im_in1*.

Sorties

- *im_out* : un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Bouchage des trous dans les régions obtenues par une simple segmentation de l'image tangram.pan image :

```
pbinarization 100 1e30 tangram.pan in.pan
pnewimage 256 256 0 255 i0.pan
psetborder 1 1 1 1 1 1 0 i0.pan i1.pan
perosionreconstruction 4 i1.pan in.pan fillhole_out.pan
```

Voir aussi

Morphologie, pdilationreconstruction.

Prototype C++

```
Errc PErosionReconstruction( const Img2duc &im_in1, const Img2duc
&im_in2, Img2duc &im_out, int connexity );
```

Auteur : Régis Clouard

peulernumberselection

Sélection de régions sur leur valeur de nombre d'Euler.

Synopsis

peulernumberselection *relation seuil* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **peulernumberselection** permet de sélectionner les régions sur leur nombre d'Euler. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

Le nombre d'Euler d'une région est:

```
euler = nombre de partie - nombre de trou.
```

Le nombre de trou d'une région peut donc être calculé à partir du nombre d'Euler = 1-E.

L'algorithme de calcul utilise une opération locale:

Soit X(R) le nombre de pattern 2x2

```
(r label de la région R, et 0 tout autre label):
0 0
0 r
```

Soit V(R) le nombre de pattern 2x2:

```
0 r
r r
```

alors Euler(R) = X(R) - V(R).

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur entière qui correspond au nombre d'Euler.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *rg_out*: une carte de régions 2D

Résultat

Retourne le nombre de régions ainsi sélectionnée.

Exemples

Sélectionne les régions avec au moins 2 trous :

(E=1-2=-1):

```
peulerselection 2 -1 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PEulerNUmberSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, long seuil );
```

Auteur: Régis Clouard

pexp

Exponentiel d'une image ou d'un graphe.

Synopsis

pexp [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pexp** construit l'exponentiel d'une image. Chaque pixel de l'image de sortie *im_out* est construit avec l'exponentiel du pixel correspondant dans l'image d'entrée *im_in*.

La formule de calcul est tout simplement:

```
pixel(im_out)=exp(pixel(im_in))
```

L'image de sortie est de type Float.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Pour les graphes, le graphe de sortie est construit avec l'exponentiel des valeurs de noeuds.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *im_out*: une image de Floats ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule l'exponentiel de l'image tangram.pan :

```
pexp tangram.pan a.pan
```

Voir aussi

Arithmetiue

Prototype C++

```
Errc PExp( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

pexponentialfiltering

Lissage par une exponentielle symétrique.

Synopsis

pexponentialfiltering *alpha* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pexponentialfiltering** permet de lisser l'image d'entrée *im_in*, par application d'un filtre exponentiel symétrique. Le filtre exponentiel est construit comme suit pour une ligne:

D'abord dans le sens du parcours causal (balayage vidéo):

```
h1[x]= alpha*(im_in[y][x]-h1[x-1]) + h1[x-1]
```

Puis le sens du parcours anti-causal (balayage anti-vidéo):

```
h2[x]= alpha*(h1[x]-h2[x+1]) + h[x+1]
```

L'opération est réitérée finalement sur les colonnes.

L'image de sortie est égale à h2[x]

```
im_out[y][x] = h2[x]
```

Paramètres

- *alpha* est une valeur réelle de l'intervalle [0,1] qui précise le degré de lissage :
  - 1 correspond à l'application d'aucun lissage,
  - 0 le plus fort lissage possible.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Effectue une détection de contours basée sur l'algorithme DOG (Différence de gaussiennes):

```
pexponentialfiltering 0.2 tangram.pan a.pan
pexponentialfiltering 0.8 tangram.pan b.pan
psub a.pan b.pan c.pan
pzerocross 8 0 c.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PExponentialFiltering( const Img2duc &im_in, Img2duc &im_out,
float alpha );
```

Auteur: Régis Clouard

pextractregion

Extraction des régions d'une image.

Synopsis

pextractregion [*rg_in*|-] [*im_in1*|-] [*im_out*|-]

Description

L'opérateur **pextractregion** construit la sous-image *im_out* qui ne contient que les pixels de l'image initiale *im_in1* contenus dans le rectangle qui englobe toutes les regions de la carte de régions *rg_in*.

Le résultat est une image de taille inférieure ou égale à l'image initiale *im_in*.

Entrées

- *rg_in*: une carte de régions.
- *im_in*: une image ou une carte de régions.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Extrait la sous-image autour des pièces de tangram puis la remet dans l'image initiale.

```
pbinarization 87 255 examples/tangram.pan a.pan
plabeling 8 a.pan b.pan
pextractregion b.pan examples/tangram.pan c.pan
pinsertregion b.pan c.pan examples/tangram.pan out.pan
```

Voir aussi

Utilitaire, pinsertregion

Prototype C++

```
Errc PExtractRegion( const Reg2d &rg_in, const Img2duc &im_in,
Img2duc &im_out );
```

Auteur: Régis Clouard

pextractsubimage

Extraction d'une sous-image d'une image.

Synopsis

pextractsubimage *x y z l h p* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pextractsubimage** construit une sous-image *im_out* avec les pixels de la fenêtre de l'image *im_in* commençant aux coordonnées (*x,y,z*) et de dimension (*l,h,p*).

Pour les cartes de régions, il n'y a pas de relabelisation, il se peut alors qu'une région se retrouve découpée en 2 et qu'il y ait des trous dans la numérotation.

Paramètres

- *x,y,z* spécifient les coordonnées du coin supérieur gauche de la fenêtre à extraire dans l'image *im_in1*.
- *l,h,p* spécifient la taille de la fenêtre à extraire. Si l'une des tailles est infeérieure à 0 ou supérieure à la taille de l'image alors c'est la taille maximale qui est utilisée.

  z et *p* ne sont pas pris en compte pour le cas des images 2D, mais doivent être donnés.

Entrées

- *im_in*: une image ou une carte de régions.

Sorties

- *im_out*: une image de même type que l'image d'entrée ou une carte de régions.

Résultat

Retourne SUCCESS ou FAILURE.
Pour les cartes de régions retourne la valeur de label maximale.

Exemples

Extrait une partie de l'image tangram.pan à partir des coordonnées 10,20 et de taille 246,236 (si tangram.pan est une 256x256).

```
pextractsubimage 10 10 0 1000 1000 0 tangram.pan a.pan
```

Voir aussi

Utilitaire, pinsertsubimage

Prototype C++

```
Errc PExtractSubImage( const Img2duc &im_in, Img2duc &im_out, Long
cx, Long cy, Long cz );
```

Auteur: Régis Clouard

pextremumsharpening

Rehaussement du contraste par utilisation des valeurs extrémales.

Synopsis

pextremumsharpening [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pextremumsharpening** effectue un réhaussement de contraste de l'image d'entrée *im_in*. L'objectif du rehaussement de contraste est de rendre plus clair les détails fins et de rendre plus nettes les parties flous. Réhausser le contraste consiste à réduire la largeur de la transition de l'intensité dans affecter l'intensité moyenne des régions de par et d'autre de la transition.

L'algorithme consiste à remplacer un pixel par la valeur minimale ou maximale de ses voisins la plus proche. Soit W un voisin, et im_in(p) un pixel de l'image d'entrée :

```
if (im_in(p)-min(W) < max-im_in(p))
then im_out[p]=min
else im_out[p]=max
```

Pour les images couleur et multispectrale, la transformation utilise l'approche marginale : l'opérateur est appliqué sur chaque bande indépendament.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image avec les mêmes propriétés que l'image d'entrée *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Réhausse le contraste de l'image tangram.pan :

```
pextremumsharpening tangram.pan a.pan
```

Voir aussi

Transformation de la LUT

Prototype C++

```
Errc PExtremumSharpening( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

pextrude1d22d

Propagation d’une valeur le long d’un axe.

Synopsis

pextrude1d22d *axis length* [*im_in*|-] [*im_out*|-]

Description

L’opérateur **pextrude1d22d**.

Paramètres

- *axis* est un entier [0..3] qui indique l’axe de projection:
  - 0: en x,
  - 1: en y,
  - 2: en z.
- *length* donne la taille de l’image pour l’axe choisi.

Entrées

- *im_in* : une image 1D.

Sorties

- *im_in* : une image 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Projection de la moyenne des lignes de l’image tangram.pan sur l’abscisse :

```
pmeanprojection 0 examples/tangram.pan a.pan
pextrude1d22d 0 256 a.pan b.pan
```

Voir aussi

Transformation

Prototype C++

```
Errc PExtrude1d22d( const Img1duc &im_in, Img2duc &im_out, int axe,
int length );
```

Auteur: Régis Clouard

pfft

Calcul de la Transformée de Fourier Rapide d'une image.

Synopsis

pfft [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out1*|-] [*im_out2*|-]

Description

L'opérateur **pfft** permet de calculer la transformée de Fourier d'une image complexe. Les images d'entrée sont:

- *im_in1* est la partie réelle de l'image.
- *im_in2* est la partie imaginaire de l'image. Si cette image n'existe pas, il faut construire une image vide (cd. psetcst).

Les images de sortie sont de type réel:

- *im_out1* est la partie réelle de la transformée.
- *im_out2* est la partie imaginaire de la transformée.

La taille des images de sortie *im_out1* et *im_out1* est calculée comme étant la puissance de 2 la plus proche de la taille des images d'entrée *im_in2* et *im_in2*.

La transformée permet de passer d'une représentation de l'image dans le domaine spatial à une représentation dans le domaine fréquentiel.

- Domaine spatial: Le domaine spatial est le domaine classique où chaque valeur en (x,y) correspond à la valeur d'intensité de la position (x',y') correspondante dans la scène observée. La distance entre deux pixels correspond à une distance réelle dans la scène.
- Domaine fréquentiel: Le domaine fréquentiel est un espace où chaque valeur de l'image à la position F représente une quantité telle que les valeurs d'intensité dans l'image I varient sur une distance spécifique relative à F. Par exemple, supposons qu'il y ait la valeur 20 au point qui représente la fréquence 0.1 (soit 1 période tous les 10 pixels), cela signifie que dans le domaine spatial de l'image correspondant, la valeur d'intensité varie du sombre au clair puis au sombre sur une distance de 10 pixels, et que le contraste entre le sombre et le clair est de 40 niveaux de gris (2 fois 20).

La transformée de Fourier représente le degré de ressemblance entre l'image vue comme une fonction f et les fonctions cosinus et sinus è différentes fréquences. Chaque point représente une fréquence particulière dans le domaine spatiale de l'image.

```
si N est le nombre de pixels total de l'image.
F(u,v) = 1/(N*N) * Sigma(x){Sigma(y){ I(x,y)*exp(-i2PI((u*i)/N+(v*i)/N))}}
```

L'équation peut être interprétée comme suit:
La valeur du point (u,v) est obtenue par la multiplication de l'image spatiale avec la fonction de base correspondante, puis addition du résultat. Les fonctions de base sont des sinus et des cosinus avec des fréquences croissantes. F(0,0) représente la composante discrète de la moyenne de l'intensité, F(N-1,N-1) représente la plus grande fréquence.

Nous ne considérons ici que la transformée discrète et donc toutes les fréquences ne sont pas prises en compte. Le nombre de fréquences considérées dépend donc de la taille de l'image. L'image de sortie *im_out* a ici la même taille que l'image d'entrée *im_in*.

Entrées

- *im_in1*: une image de niveaux de gris (la partie réelle de la transformation).
- *im_in2*: une image de niveaux de gris (la partie imaginaire de la transformation).

Sorties

- *im_out1*: une image de niveaux de gris (la partie réelle de la transformation).
- *im_out2*: une image de niveaux de gris (la partie imaginaire de la transformation).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule l'amplitude de la transformée de Fourier de l'image tangram.pan. La partie imaginaire (i1.pan) est nulle. (Uitiliser la transformation log de pvisu pour visualiser le résultat out.pan.):

```
psetcst 0 tangram.pan i1.pan
pfft tangram.pan i1.pan i2.pan i3.pan
pfftshift i2.pan i3.pan i4.pan i5.pan
pmodulus i4.pan i5.pan out.pan
```

Voir aussi

Domaine Fréquentiel, pifft, pfftshift

Prototype C++

```
Errc PFFT( const Img2duc &im_in1, const Img2duc &im_in2, Img2dsf
&im_out1, Img2dsf &im_out2 );
```

Auteur: Herissay & Berthet

pfftconvolution

Convolution d'une image par un noyau.

Synopsis

pfftconvolution [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pfftconvolution** permet de convoluer l'image initiale *im_in1* par le noyau donné dans l'image *im-in2*. Cette convolution utilise le passage par le domaine frequentiel avec la transformée de Fourier.

La taille du noyau doit être inférieure à celle de l'image d'entrée. Le noyau doit être centrée dans l'image du noyau.

La convolution consiste donc en une multiplication complexe des transformées de Fourier des deux images d'entrée, soit la séquence:

- imi=fft(im_in) * fft(im_in2)
- im_out=iift(imi);

où * est la multiplication complexe.

Entrées

- *im_in1*: une image 2D.
- *im_in2*: une image 2D (de même type que l'image d'entrée *im_in1* et d'une taille inférieure ou ègale à *im_in1*).

Sorties

- *im_out*: une image réelle de même taille que *im_in1*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Ajoute du flou de bougé dans l'image tangram.pan Le flou de bougé est généré par l'intermédiaire d'une ligne oblique:

```
pshapedesign 10 10 0 3 10 1 line.pan
protation 0 45 line.pan line1.pan
pfftconvolution tangram.pan line1.pan out.pan
```

Voir aussi

Domaine Fréquentiel, pconvolution, pftt, piftt

Prototype C++

```
Errc FFTConvolution( const mg2duc &im_in1, const Img2duc &im_in2,
Img2dsf &im_out );
```

Auteur: Régis Clouard

pfftcorrelation

Corrélation entre deux images.

Synopsis

pfftcorrelation [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pfftcorrelation** permet de calculer la correlation entre deux image *im_in1* et *im_in2*.

La corrélation permet de déterminer un degré de ressemblance entre deux images.

Cette corrélation utilise le passage par le domaine frequentiel avec la transformée de Fourier.

Les deux images doivent être de même type et de même taille.

La corrélation consiste donc en une multiplication complexe entre la transformée de Fourier de l'image *im_in1* et le complexe conjugué de la transformée de Fourier de l'image *im_in2*, soit la séquence:

- imi=fft(im_in) * conj(fft(im_in2))
- im_out=iift(imi);

où * est la multiplication complexe et conj(im) le complexe conjugué.

Entrées

- *im_in1*: une image 2D.
- *im_in2*: même type et même taille que *im_in1*.

Sorties

- *im_out*: une image de Float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Detecte la présence de pièces de tangram dans l'image tangram.pan:

```
pextractsubimage 111 6 0 35 32 0 tangram.pan a.pan
pfftcorrelation tangram.pan a.pan b.pan
plocalmaxima 8 b.pan out.pan
```

Voir aussi

Domaine Fréquentiel, pftt, piftt

Prototype C++

```
Errc PFFTCorrelation( const Img2duc &im_in1, copnst Img2duc &im_in2,
Img2dsf &im_out );
```

Auteur: Régis Clouard

pfftdeconvolution

Déconvolution d'une image par un noyau.

Synopsis

pfftdeconvolution [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pfftdeconvolution** permet de déconvoluer l'image initiale *im_in1* par le noyau donné dans l'image *im-in2*. Cette deconvolution utilise le passage par le domaine frequentiel avec la transformée de Fourier.

La taille du noyau doit être inférieure à celle de l'image d'entrée.

La deconvolution consiste donc en une division complexe des transformées de Fourier des deux images d'entrée, soit la séquence :

- imi=fft(im_in) / fft(im_in2)
- im_out=iift(imi);

où / est la division complexe.

Entrées

- *im_in1*: une image 2D.
- *im_in2*: une image 2D (de même type que *im_in1* et de taille inférieure ou égale à *im_in1*).

Sorties

- *im_out*: une image réelle de même taille que *im_in1*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Voir aussi

Domaine Fréquentiel, ftt, iftt

Prototype C++

```
Errc PFFTDeconvolution( const Img2duc &im_in1, const Img2duc
&im_in2, Img2dsf &im_in2 );
```

Auteur: Régis Clouard

pfftshift

Permutation des 4 sous-images de la transformée de Fourier.

Synopsis

pfftshift [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out1*|-] [*im_out2*|-]

Description

L'opérateur **pfftshift** permet de permuter 2 à 2 les 4 sous-images des 2 images (partie réelle et partie imaginaire) issues de la transformée de Fourier.

Ainsi les cadrans 1,2 3,4 sont permutés en 4,3,2,1

```
+--+--+ +--+--+
|1 |2 | |4 |3 |
+--+--+ +--+--+
|3 |4 | |2 |1 |
+--+--+ +--+--+
```

Entrées

- *im_in1*: une image 2D de réels.
- *im_in2*: une image 2D de réels.

Sorties

- *im_out1*: une image 2D de réels.
- *im_out2*: une image 2D de réels.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule l'amplitude de la transformée de Fourier de l'image tangram.pan. La partie imaginaire (i1.pan) est nulle. (Uitiliser la transformation log de pvisu pour visualiser le résultat out.pan.):

```
psetcst 0 tangram.pan i1.pan
pfft tangram.pan i1.pan i2.pan i3.pan
pfftshift i2.pan i3.pan i4.pan i5.pan
pmodulus i4.pan i5.pan out.pan
```

Voir aussi

Domaine Fréquentiel

Prototype C++

```
Errc PFFTShift( const Img2dsf &im_in1, Img2dsf &im_in2, Img2dsf
&im_out1, Img2dsf &im_out2 );
```

Auteur: Régis Clouard

pfile

Présentation des propriétés d’un fichier Pandore.

Synopsis

pfile *im_in*

Description

L’opérateur **pfile** permet d’afficher la description des propriétés du fichier Pandore donné en paramètre.

Par exemple, appliqué sur une image, **pfile** donne le nom du créateur, la date de création, le type des pixels, le nombre de lignes, le nombre de colonnes, etc.

Entrées

- *im_in*: une image, une carte de régions, un graphe ou une collection.

Résultat

Pas de valeur de retour.

Exemples

- Affiche les informations de l’image "tangram.pan" :

```
pfile examples/tangram.pan
```

Voir aussi

Information

Auteur: Régis Clouard

pfillhole

Bouchage des trous dans les régions.

Synopsis

pfillhole *connexity* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pfillhole** construit la carte de régions de sortie *rg_out* avec les régions de la carte de régions *rg_in* auxquelles les trous intérieurs ont été comblés.

Un **trou** est défini comme une région du fond (label=0) qui n'est entourée que d'une seule autre région. C'est la *connexity* qui définit la notion de connexité d'un trou. Si la continuité des trous est définie en 4 connexité alors la continuité des régions est définie en 8 connexité et inversement. Une région du fond qui touche un bord n'est pas considérée comme un trou (ceci pour éviter qu'une image ne contenant que du fond se retrouve bouchée).

Les trous sont agglomérés dans la région de *rg_in* qui l'entoure.

En sortie, *rg_out* conserve le même nombre de labels que *rg_in* et les mêmes valeurs de label pour les régions.

Paramètres

- *connexity* spécifie la connexité d'un trou (4 ou 8 pour les images 2D image - 6 ou 26 pour les images 3D).

Entrées

- *rg_in*: une carte de régions.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de trous bouchés.

Exemples

Bouche les trous (4-connexes) des régions de la carte de régions rin.pan:

```
pfillhole 4 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PFillHole( const Reg2d &rg_in, Reg2d &rg_out, int connexity );
```

Auteur: Régis Clouard

pfisher

Multiseuillage de l'image par partitionnement de l'histogramme des niveaux de gris.

Synopsis

`Pfisher` *`minval nbclass`* [*`-m mask`*] [*`im_in`*|`-`] [*`im_out`*|`-`]

Description

L'opérateur **Pfisher** permet de faire une classification des pixels de l'image *im_in* en *nbclass* classes, en utilisant le partitionnement de l'histogramme en *nbclass* classes disjointes tel que la somme des variances des classes soit minimale.

L'image de sortie *im_out* est contruite avec les seuils détectés, telle que:

`im_out[y][x]=seuil[k] si seuil[k-1]<im_out[y][x]<=seuil[k].`

Le dernier seuil est égal à la valeur maximale 255.

Remarque: Cet opérateur ne fonctionne que sur des images de Char parce qu'il faut que les histogrammes soient suffisamment significatifs. Il faut donc s'arranger pour transformer les autres types d'images en image de Uchar.

Paramètres

- *minval* est la valeur de niveau de gris minimale à partir de laquelle est construit l'histogramme. Ce paramètre vaut en général 0, mais vaut au moins à 1 lorsque l'on utilise un masque.
- *nbclass* spécifie le nombre de classes souhaitées en sortie. Ce nombre est un entier compris entre 2 et 25.

Entrées

- *im_in*: une image 2D d'octets (Img2duc, Img3duc).

Sorties

- *im_out*: une image 2D d'octets (Img2duc, Img3duc).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Segmente les pièces de tangram:

```
pfisher 0 2 tangram.pan out.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PFisher( const Img2duc &im_in, Img2duc &im_out, Uchar minval,
int nbclass );
```

Auteur: Régis Clouard

pfits2pan

Conversion d'une image au format FITS vers le format Pandore.

Synopsis

pfits2pan [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pfits2pan** convertie une image au format FITS (Flexible Image Transport System) en image Pandore.

Seules les images de niveaux de gris sont considérées ici. L'image de sortie est toujours de type Float.

Entrées

- *im_in*: un fichier de type FITS.

Sorties

- *im_out*: une image Pandore.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image au format fits en image au format Pandore:

```
pfits2pan image.fits image.pan
```

Voir aussi

Conversion, ppan2fits

Prototype C++

```
Pobject* Fits2Pan( const char *filename );
```

Auteur: Jalal Fadili

pflip

Construction du symétrique d'une image.

Synopsis

pflip *axis* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pflip** permet d'obtenir une image symétrique (ou image miroir) de l'image d'entrée *im_in* selon l'une des trois directions possibles, en x en y ou en z.

Par exemple, la symétrie en x signifie que l'image de sortie est construite par:

```
im_out[y][x]=im_in[y][width-x-1]
```

Paramètres

- *axis* donne l'axe de symétrie:
  - 0 = symétrie sur l'axe x.
  - 1 = symétrie sur l'axe y.
  - 2 = symétrie sur l'axe z.

Entrées

- *im_in*: une image ou une carte de régions.

Sorties

- *im_in*: une image de même type que l'image d'entrée ou une carte de régions.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit le symétrique de l'image tangram.pan:

```
pflip 0 tangram.pan a.pan
```

Voir aussi

Transformation

Prototype C++

```
Errc PFlip( const Img3duc &im_in,Img3duc &im_out, int axis );
```

Auteur: François Angot

pfuzzyclustering

Classification des pixels d'une image par la méthode des k moyennes floues.

Synopsis

pfuzzyclustering *nbclass degre_flou* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pfuzzyclustering** permet de faire une classification des pixels de l'image *im_in* en *nbclass* classes. La méthode utilise l'algorithme des C moyennes floues.

La carte de régions de sortie *rg_out* est faite avec des labels croissants pour chacune des classes extraites de l'image.

Paramètres

- *nbclass* indique le nombre de classes disjointes désirées en sortie.
- *degre_flou* permet de définir le degré de flou de la méthode. C'est un réel entre 1 et 2: 1 pas de flou, 2 flou maximum.

Entrées

- *im_in*: une image 2D.

Sorties

- *rg_out*: une carte de régions 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Segmente les pièces de tangram:

```
pfuzzyclustering 2 1.5 tangram.pan out.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PFuzzyClustering( const Img2duc &im_in, Reg2d &rg_out, int
nbclass, float degre_flou );
```

Auteur: Jalal Fadili

pgaussaggregation

Croissance des régions d'une carte selon une distribution gaussienne.

Synopsis

pgaussaggregation *connexite alpha* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pgaussaggregation** consiste à agglomérer des pixels à une région connexe lorsque sa valeur de pixel est proche de celle de la région, c'est à dire que sa valeur appartient à l'intervalle:

```
[m(R) - alpha*s(R), m(R) + alpha*s(R)],
où est m(R) est la moyenne intérieure et s(R) l'écart-type de la région R.
```

Les pixels à agglomérer sont les pixels non encore étiquétés dans la carte de régions *rg_in* (ceux qui ont un label=0).

La moyenne et l'écart type des régions de *rg_in* ne sont pas recalculées pour éviter de trop s'éloigner de la situation initiale. On préferera des exécutions itératives de cet opérateur. On pourra pour exemple itérer cet opérateur jusqu'à ce que le résultat de pstatus = 0. Ainsi, à chaque appel de l'opérateur la moyenne et l'écart-type sont recalculé. avec les nouvelles régions.

L'utilisation de cet opérateur n'a de sens que si les régions germes ont suffisamment de pixels pour construire une gaussienne significative (un écart-type non nul).

La carte de sortie *rg_out* a le même nombre de labels que la carte d'entrée *rg_in*.

Paramètres

- La *connexite* possible est liée à la dimension de l'image: 4 ou 8 pour le 2D, et 6 ou 26 pour le 3D.
- *alpha* est une valeur réelle qui définit la taille de l'intervalle d'acceptation (valeur typique 1).

Entrées

- *rg_in*: une carte de régions.
- *im_in*: une image de niveaux de gris.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre total de pixels qui ont été aggrégés à une région. Retourne FAILURE en cas de problème.

Exemples

Aggrège les pixels des pièces de tangram :

```
pbinarization 96 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pgaussaggregation 8 4 b.pan tangram.pan out.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PGaussAggregation( const Reg2d &rg_in, const Img2duc &im_in,
Reg2d &rg_out, int connexite, float alpha );
```

Auteur: Régis Clouard

pgaussclassification

Classification utilisant un modèle gaussien.

Synopsis

pgaussclassification *attr_base attr_in attr_out* [*col_base*|-] [*col_in*|-] [*col_out*|-]

Description

L'opérateur `pgaussclassification` implémente une classification basée sur un modèle gaussien. L'idée. est de modéliser la distribution de chaque classe par une gaussienne, puis pour un x donné de rechercher la classe qui maximise la probabilité de contenir x. Pratiquement, `pgaussclassification` cherche la classe `i` qui minimise:

$$f(x,i) = \ln(\det A(i)) + {}^{t}(x - m(i)).A(i)^{-1}.(x - m(i)) - \ln(P(i)^{2})$$

- `x` est le vecteur de caractéristiques représentant l' à classifier ;
- `A(i)` la matrice de covariance associée à la classe `i` ;
- `m(i)` le vecteur de moyennes des caractéristiques de la classe `i` ;
- `P(i)` la probabilité *a priori* de trouver la classe `i`.

Paramètres

- La collection `col_base` doit contenir les paramètres de la formule ci-dessus.
  Si l'on suppose qu'il existe n classes et p caractéristiques par élément :
  - `attr_base.moy` est un tableau de `n*p` flottants contenant à l'indice [`i*n+j`] la moyenne de la (`j+1`)ième caractéristique de la (`i+1`) classe.
  - `attr_base.det` est un tableau de n flottants contenant à l'indice [`i-1`] la valeur de `det(A(i))`.
  - `attr_base.inv` est le tableau des matrices `A^(-1)`; l'indice [`k*p*p + i*p +j`] correspond à la cellule `(i+1,j+1)` de la `(k+1)`ième matrice.

  (Ces trois 3 attributs peuvent être calculés avec l'opérateur parraycovarmat.)
  - `attr_base.pap` est le tableau des probabilités *a priori* pour chaque classe (un tableau de `n` flottants donc).

  (Ce dernier attribut peut-être omis, dans le cas où les classes sont équiprobables.)
- La collection `col_in` doit contenir les `x` à classifier, sous la forme d'attributs `attr_in.1`, `attr_in.2`, ..., `attr_in.p`,
  qui sont des tableaux contenant les caractéristiques de chaque élément à classifier.
- La collection `col_out` contient un tableau de Ushort, donnant la classe déterminée pour chaque de `col_in`.

Entrées

- *col_base*: une collection contenant les paramètres appris.
- *col_in*: une collection à classer.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Classification des bonbons de l'image jellybean.pan à partir d'un échantillon des différents types de bonbons stockés dans le dossier 'base' (Unix version):

```
# Learning
classes=1
for i in base/*.pan
do
    pim2array ind $i /tmp/tmp1
    parray2array ind.1 Float /tmp/tmp1| parray2array ind.2 Float | parray2array ind.3 Float - a.pan
    parraycovarmat ind ind a.pan i-01.pan
    if [ -f base.pan ]
    then pcolcatenateitem i-01.pan base.pan base.pan
    else cp i-01.pan base.pan
    fi
    classe=‘expr $classe + 1‘
done
rm /tmp/tmp1

# Classification
pim2array ind jellybeans.pan a.pan
parray2array ind.1 Float a.pan| parray2array ind.2 Float | parray2array ind.3 Float - b.pan
pgaussclassification ind ind ind base.pan b.pan | parray2im $ncol $nrow 0 ind | pim2rg - out.pan
```

Voir aussi

Classification

Prototype C++

```
Errc PGaussClassification(const std::string &a_base, const
Collection &c_base, const std::string &a_in, const Collection &c_in,
const std::string &a_out, Collection &c_out);
```

Auteur: Alexandre Duret-Lutz

pgaussianfilter

Génère un filtre Gaussien passe-bas, passe-haut, coupe-bande ou passe-bande.

Synopsis

pgaussianfilter [-m mask] *ncol nrow ndep highpass cutin cutoff* [*im_out*|-]

Description

L'opérateur **pgaussianfilter** génère un filtre Gaussien passe-bas, passe-haut, coupe-bande ou passe-bande.

Si *ndep<1* le filtre *im_out* est une image 2D de Float avec size nrow*ncol sinon le filtre *im_out* est une image 3D de Float avec la taille ndep*nrow*ncol.

Le filtre passe-bas de Butterworth coupe les hautes fréquences des composantes de la transformée de Fourier qui sont à une distance supérieure à la distance spécifiée D0 (la valeur *cutoff*) à partir de l'origine du centre de la transformation.

La fonction de transfert d'un filtre 2D passe-bas gaussien avec une fréquence de coupe à la distance D0 de l'origine est définie par :

$$Hlp(u,v) = \exp(-D^2(u,v)/2D0^2)$$

où D(u,v) est la distance du point (u,v) à l'origine :

$$D(u,v)=\mathrm{sqrt}((u-M/2)^2 + (v-N/2)^2)$$

où N est le nombre de lignes et M est le nombre de colonnes.

La fonction de transfert d'un filtre passe-haut :

$$H(u,v) = 1- Hlp(u,v)$$

La fonction de transfert d'un filtre coupe-bande :

$$H(u,v) = Hhp(u,v) - Hlp(u,v)$$

où Hhp(u,v) est le filtre passe-haut avec le paramètre cutoff et Hlp(u,v) est le filtre passe-bas avec le paramètre cutin.

Paramètres

- *ncol*, *nrow*, *ndep* spécifie la taille de l'image de sortie. Si *ndep*<1 alors la sortie est une image 2D sinon une image 3D.
- *highpass* est utilisée. conjonction avec le paramètre *cutin*. Il spécifie le type de filtre :
    - highpass=0 et cutin=0 : filtre passe-bas
    - highpass=1 et cutin=0 : filtre passe-haut
    - highpass=0 et cutin=1 : filtre coupe-bande
    - highpass=1 et cutin=1 : filtre passe-bande.
- *cutin* est la fréquence de coupe du filtre D0 en cas de filtre coupe-bande ou bande-passe. Dans ce cas, l'épaisseur des bandes = cutoff-cutin et D0=(cutoff+cutin)/2.
- *cutoff* est la fréquence de coupe du filtre D0. C'est un réel positif dans l'intervalle]0.. sqrt(M*m+N*n)/2]. Il correspond à la distance euclidienne de la bande au centre de l'image. Plus cutoff est élevé, plus le filtrage passe-bas est faible et plus le filtrage passe_haut est fort.

Sorties

- *im_out*: une image de Float (Img2dsf).

Résultat

Retourne SUCCESS ou FAILURE en cas de mauvais paramétrage.

Exemples

Effectue un filtrage passe-bas de Gauss :

```
pimage1 0 tangram.pan i1.pan
pfft tangram.pan i1.pan i2.pan i3.pan
pgaussianfilter 256 256 0 0 0 100 i4.pan
pmult i2.pan i4.pan i5.pan
pmult i3.pan i4.pan i6.pan
pifft i5.pan i6.pan out.pan i8.pan
```

Effectue un filtrage passe-haut de Gauss :

```
psetcst 0 tangram.pan i1.pan
pfft tangram.pan i1.pan i2.pan i3.pan
pgaussianfilter 256 256 0 1 0 50 i4.pan
pmult i2.pan i4.pan i5.pan
pmult i3.pan i4.pan i6.pan
pifft i5.pan i6.pan out.pan i8.pan
```

Effectue un filtrage coupe-bande de Gauss :

```
psetcst 0 tangram.pan i1.pan
pfft tangram.pan i1.pan i2.pan i3.pan
pgaussianfilter 256 256 0 0 25 50 i4.pan
pmult i2.pan i4.pan i5.pan
pmult i3.pan i4.pan i6.pan
pifft i5.pan i6.pan out.pan i8.pan
```

Effectue un filtrage passe-bande de Gauss :

```
psetcst 0 tangram.pan i1.pan
pfft tangram.pan i1.pan i2.pan i3.pan
pgaussianfilter 256 256 0 1 25 50 i4.pan
pmult i2.pan i4.pan i5.pan
pmult i3.pan i4.pan i6.pan
pifft i5.pan i6.pan out.pan i8.pan
```

Voir aussi

Domaine Fréquentiel, pifft, pfftshift

Prototype C++

```
Errc GaussianFilter( Img2dsf &im_out, int ndep, int nrow, int ncol,
int highpass, float cutin, float cutoff );
```

Auteur: Régis Clouard

pgaussianfiltering

Lissage d'une image par une gaussienne.

Synopsis

pgaussianfiltering *sigma* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

Le lissage de **pgaussianfiltering** consiste à appliquer sur les lignes puis les colonnes un filtre de Gauss de taille (sigma*6). Ce filtre définit l'influence des pixels voisins sur la valeur du pixel central. C'est un filtre moyenneur spatial.

Le filtre F(i) de taille (6*sigma) est construit comme suit:

```
F(i)= exp(-((Double)(i-demitaille)*(i-demitaille)/(2.0*sigma*sigma)))
```

avec demitaille = sigma*3.

Le bord est traité par recopie du pixel du bord.

Paramètres

- *sigma* est l'écart type de la gaussienne. Il permet aussi de déterminer la taille du filtre (6*sigma). C'est un réel compris entre [0 et tailleimage/6]. Plus *sigma* est élevé plus le lissage est fort. Il est généralement lié à la taille des objets présents dans l'image: si les objets sont petits il vaut mieux un lissage faible de manière à ne pas perdre les objets.

Entrées

- *im_out*: une images.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applique un filtrage gaussien à l'image tangram.pan:

```
pgaussianfiltering 1 tangram.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PGaussianFiltering( const Img2duc &im_in, Img2duc &im_out,
float sigma );
```

Auteur: Régis Clouard

pgeodesicdilation

Dilatation géodésique des points de plus fort contraste de l'image.

Synopsis

pgeodesicdilation *num_se halfsize iteration* [*-m mask*] [*im_in*|-] [*im_msq*|-] [*im_out*|-]

Description

L'opérateur **pgeodesicdilation** effectue la dilatation des pixels de l'image *im_in* tant que ceux-ci appartiennent à une zone non nulle spécifiée dans l'image *im_msq*. *im_msq* est une image de Char ou une carte de régions utilisée comme masque binaire. Tous les pixels non nuls correspondent à une valeur vrai pour le masque. L'élément structurant est choisi parmi une liste par *num_se* et sa demi-taille *halfsize*.

La dilatation géodésique correspond à l'opération:

```
if im_msq (p)!=0
   dilatation(p) = MAX(neighbors of p specified by the structuring element)
else
   dilatation(p) = im_in(p).
```

La dilatation conditionnelle s'écrit comme:

```
pdilation se hs in.pan i1.pan
pmask i1.pan msq.pan out.pan
```

Si *iteration=-1* l'opération est appliquée jusqu'à idempotence.

Pour les cartes de régions, la dilatation s'effectue uniquement à l'intérieur d'une même région.

Pour les images couleur, c'est l'ordre lexicographique qui est utilisé : d'abord en utilisant la bande X, en cas d'égalité en utilisant la bande Y puis la bande Z.

Paramètres

- *num_se* spécifie le type de l'élément structurant :

  En 2D:

  - 0: losange (4-connexité)
  - 1: carré (8-connexité).
  - 2: cercle
  - 3: ligne horizontale (-)
  - 4: ligne verticale (|)

- 5: ligne oblique droire (/).
- 6: ligne oblique gauche (\).

En 3D:

- 0: bipyramide (6-connexité)
- 1: cube (26-connexitelle sé)
- 2: sphère
- 3: ligne en x (-)
- 4: ligne en y (|)
- 5: ligne en z

- *halfsize* donne la demi-taille de l'élément structurant. Par exemple, une demi-taille de 1 pour un carré donne un élément structurant de taille 3x3.

- *iteration* est un entier positif qui donne le nombre de dilatation géodésique à opérer. Si *iteration* vaut -1, alors la dilatation géodésique est effectuée jusqu'à idempotence.

Entrées

- *im_in*: une image de niveaux de gris ou une image couleur.
- *im_msq*: une image d'octets ou une carte de régions.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Détection de contours basée sur le seuillage par hystéresis.

```
pgradient 1 tangram.pan i1.pan i2.pan
pbinarization 80 -1 i1.pan i3.pan
pbinarization 30 -1 i1.pan i4.pan
pgeodesicdilation 1 1 -1 i3.pan i4.pan out.pan
```

Voir aussi

Morphologie, pgeodesicerosion

Prototype C++

```
Errc PGeodesicDilation( const Img2duc &im_in, const Img2duc &im_msq,
Img2duc &im_out, int num_se, int halfsize, int iteration );
```

Auteur: Régis Clouard

pgeodesicerosion

Erosion géodésique des points de plus fort contraste de l'image.

Synopsis

pgeodesicerosion *num_se halfsize iteration* [*-m mask*] [*im_in*|-] [*im_msq*|-] [*im_out*|-]

Description

L'opérateur **pgeodesicerosion** effectue l'érosion des pixels de l'image *im_in* tant que ceux-ci appartiennent à une zone nulle spécifiée dans l'image *im_msq*. *im_msq* est une image de char ou une carte de régions utilisée comme masque binaire. Tous les pixels non nuls correspondent à une valeur vrai pour le masque. L'élément structurant est choisi parmi une liste par *num_se* et sa demi-taille *halfsize*.

L'érosion géodésique correspond à l'opération:

```
if im_msq (p)=0
   erosion(p) = MIN(voisins selon l'élément structurant de x,y)
else
   erosion(p) = im_in(p).
```

L'érosion conditionnelle s'écrit comme:

```
pinverse msq.pan i1.pan
por in.pan i1.pan i2.pan
perosion hs i2.pan i3.pan
pmask i3.pan msq.pan out.pan
```

Si *iteration=-1*, l'opération est appliquée jusqu'à idempotence.

Pour une image binaire cela revient à éroder les régions blanches.

Pour les cartes de régions, l'érosion ajoute des pixels de label=0 aux points d'érosion.

Pour les images couleur, c'est l'ordre lexicographique qui est utilisé : d'abord en utilisant la bande X, en cas d'égalité en utilisant la bande Y puis la bande Z.

Paramètres

- *num_se* spécifie le type de l'élément structurant :

  En 2D:

    - 0: losange (4-connexité)
    - 1: carré (8-connexité).
    - 2: cercle
    - 3: ligne horizontale (-)
    - 4: ligne verticale (|)
    - 5: ligne oblique droire (/).
    - 6: ligne oblique gauche (\).

    En 3D:

    - 0: bipyramide (6-connexité)
    - 1: cube (26-connexité)
    - 2: sphère
    - 3: ligne en x (-)
    - 4: ligne en y (|)
    - 5: ligne en z
- *halfsize* donne la demi-taille de l'élément structurant. Par exemple, une demi-taille de 1 pour un carré donne un élément structurant de taille 3x3.

- *iteration* est un entier positif qui donne le nombre d'érosion géodésique à opérer. Si *iteration* vaut -1, alors l'érosion géodésique est effectuée jusqu'à idempotence.

Entrées

- *im_in*: une image couleur ou en niveaux de gris.
- *im_msq*: une image d'octets ou une carte de régions.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Détection de contours basée sur le seuillage par hystéresis.

```
pgradient 1 tangram.pan i1.pan i2.pan
pbinarization 0 80 i1.pan i3.pan
pbinarization 0 30 i1.pan i4.pan
pgeodesicerosion 1 1 -1 i3.pan i4.pan i5.pan
pinverse i5.pan out.pan
```

Voir aussi

Morphologie, pgeodesicdilation

Prototype C++

```
Errc PGeodesicErosion( const Img2duc &im_in, const Img2duc &im_msq,
Img2duc &im_out, int num_se, int halfsize, int iteration );
```

Auteur: Régis Clouard

pgetband

Récupération une bande dans une image multispectrale.

Synopsis

pgetband *band* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **psetband** permet de créer une nouvelle image couleur ou multispectrale où l'image *im_in1* remplace la bande *band* de l'mage *im_in2*.

Paramètres

- *band* est un entier. Si sa valeur est supérieure ou inférieure au nombre de bandes de l'image d'entrée alors la bande la plus proche est utilisée (première ou dernière bande).

Entrées

- *im_in1*: une image.
- *im_in2*: une image multispectrale.

Sorties

- *im_out*: une image multispectrale.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Extrait la première bande de l'image couleur parrot.pan:

```
pgetband 0 parrot.pan a.pan
```

Voir aussi

Utilitaire, psetband.

Prototype C++

```
Errc PGetBand( const Imc2duc &ims, const Img2duc &ims2, int band);
```

Auteur: Régis Clouard

pgetquadrangle

Sélection du meilleur quadrilatère dans un ensemble de lignes.

Synopsis

pgetquadrangle [*-m mask*] [*im_lines*|-] [*im_bin*|-] [*im_out*|-]

Description

L'opérateur **pgetquadrangle** permet de détecter le meilleur quadrilatère parmi les lignes de l'image *im_lines*. Détecter le tableau blanc consiste à repérer Seul est retenu le quadrilatère qui satisfait le mieux aux contraintes suivantes :

- Les lignes opposées doivent avoir une orientation proche (+/- 30°).
- Les lignes opposées doivent être suffisamment loin l'une de l'autre (distance > 1/5 de la largeur ou de la hauteur).
- L'angle entre deux lignes voisines doit être proche de 90° (+/- 30°).
- Le quadrilatère doit être suffisamment grand (de périmètre > (W + H)/4).
- Le quadrilatère choisi est celui qui a le meilleur rapport entre le périmètre et la nombre de points de l'image binaire *im_bin* sous les bords du quadrilatère d'une épaisseur de 3 pixels.

Entrées

- *im_lines*: une image binaire contenant les lignes droites (généralement issues de phouglines).
- *im_bin*: une image binaire contenant les contours (généralement issues d'une binarization d'une image de gradient).

Résultat

FAILURE s'il n'existe aucun quadrilatère possible.

Exemples

Détection des bords du tableau blanc :

```
pshen 1.3 whiteboard.pan a.pan
pbinarization 2 255 a.pan bin.pan
phoughlines 10 0 360 a.pan lines.pan
pgetquadrangle lines.pan a.pan result.pan
```

Voir aussi

Reconstruction

Prototype C++

```
Errc PGetQuadrangle( const Img2duc &im1_in, const Img2duc &im2_in,
Img2duc &im_out );
```

Reference

Z. Zhang, and L. He, "*Whiteboard Scanning and Image Enhancement*", Digital Signal Processing, Vol.17, No.2, pages 414-432, 2007.

Auteur: Régis Clouard

pgetslice

Construction d'une image 2D avec un plan d'une image 3D.

Synopsis

pgetslice *slice* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pgetslice** construit une image 2D *im_out* à partir d'un plan de l'image 3Dd *im_in*.

L'image résultante *im_out* est du même type que l'image d'entrée.

Paramètres

- *slice* spécifie l'indice du plan à récupérer. C'est un entier entre 0 et le nombre de plan total de l'image *im_in* moins 1. Si *slice*=0 ou > nombre total alors c'est le dernier plan qui est extrait.

Entrées

- *im_in*: une image 3D ou une carte de régions 3D.

Sorties

- *im_out*: une image 2D du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Récupère le plan no. 10 de l'image 3D (le 11e plan) de l'image a3d.pan:

```
pgetslice 10 a3d.pan a2d.pan
```

Voir aussi

Utilitaire, psetslice, paddslice

Prototype C++

```
Errc PGetSlice( const Img3duc &im_in, Img2duc &im_out, long slice );
```

Auteur: Régis Clouard

pgetsubband

Extraction d'une sous-bande d'une image de DWT.

Synopsis

pgetsubband *scale subband* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pgetsubband** récupère une sous-bande d'une image DWT à l'échelle spécifiée *scale*. La sous-bande *im_out* est une image.

A chaque échelle, les images sont numérotée ainsi:

```
[1][2]
[3][4]
```

- 1: sous-bande LL des coefficients d'approximation.
- 2: sous-bande LH des coefficients de détail.
- 3: sous-bande HL des coefficients de détail.
- 4: sous-bande HH des coefficients de détail.

Paramètres

- *scale* spécifie l'échelle d'analyse de l'image DWT.
- *subband* spécifie le numéro de la sous-bande à récupérer à l'échelle donnée.

Entrées

- *im_in*: une image 2D de niveaux de gris.

Entrées

- *im_out*: une image du même type que l'image d'entree.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Récupère les images LL et LH de l'analyse par ondelette d'une image d'un carré.

```
pshapedesign 256 256 0 2 150 150 a.pan
pqmf daubechies 4 b.pan
pdwt 1 a.pan b.pan c.pan
pgetsubband 1 1 c.pan out1.pan
pgetsubband 1 2 c.pan out2.pan
```

Voir aussi

Domaine Fréquentiel

Prototype C++

```
Errc PGetSubband( const Img2dsf &im_in, Img2dsf &im_out, int scale,
int subband );
```

Auteur: Ludovic Soltys

pgetwindowaroundpoints

Extraction des pixels dans la fenêtres autour de points spécifiés.

Synopsis

pgetwindowaroundpoints *ncol nrow ndep* [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pgetwindowaroundpoints** construit une image avec les pixels de *im_in1* qui se situent dans une fenêtre de taille ncol*nrow*ndep autour des points donnés dans l'image *im_in2*.

L'image résultante *im_out* est du même type que l'image d'entrée *im_in2*.

Paramètres

- *ncol*, *nrow*, *ndep* spécifient la taille de la fenêtre de récupération des pixels.

Entrées

- *im_in1*: une image.
- *im_in2*: une image binaire avec les points.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Extrait les fenêtres autour des points d'intérêt des pièces de tangram.

```
pharris 2 0.04 examples/tangram.pan a.pan
pbinarization 1e4 1e30 a.pan b.pan
pgetwindowaroundpoints 10 10 0 examples/tangram.pan b.pan result.pan
```

Voir aussi

Utilitaire,

Prototype C++

```
Errc PGetWindowAroundPoints( const Img2duc &ims1, const Img2duc
&ims2, Img2duc &imd, int h, int w );
```

Auteur: Régis Clouard

pgif2pan

Conversion d'une image GIF en image Pandore.

Synopsis

pgif2pan *im_in* [*im_out*|-]

Description

L'opérateur **pgif2pan** permet de convertir une image de type *GIF* en un fichier *Pandore*. Le fichier résultant est de type:

- Img2duc pour les images en niveaux de gris,
- Imc2duc pour les images couleur.

Remarque: Seul les fichiers GIF non compressés sont pris en compte. Il peut alors être nécessaire d'utiliser un autre logiciel de conversion pour supprimer la compression.

Entrées

- *im_in*: un fichier GIF.

Sorties

- *im_out*: une image Pandore.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pgif2pan image.gif image.pan
```

Voir aussi

Conversion, ppan2gif

Prototype C++

```
Errc PGif2Pan( const FILE* fdin, Pobject **objout );
```

Auteur: Régis Clouard

pgr2im

Conversion d’un graphe en une image.

Synopsis

pgr2im *type* [*gr_in*|-] [*im_out*|-]

Description

L’opérateur **pgr2im** construit une image *im_out* qui fournit une représentation visuelle du graphe *gr_in*. Pour chaque sommet, si le champ "state" ne contient pas 0, le point correspondant est dessiné sur l’image de sortie.

Les arcs à dessiner sont définis par le *type* de représentation.

L’image de sortie *im_out* est de type float, puisque le champ "state" est de type float.

Paramètres

- *type* permet choisir les arcs à afficher:
  - 1- Les arcs entre deux sommets de valeurs non nulles.
  - 2- Les arcs joignant deux sommets de même valeur "state".
  - 3- Les arcs partant de sommets dont le champ "value" est non nul.

Entrées

- *gr_in*: un graphe.

Sorties

- *im_out*: une image de type float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Consstruit une image avec le graphe g.pan:

```
pgr2im 0 a.pan b.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PGr2Im( const Graph2d &grs, Img2dsf &imd, int type );
```

Auteur: Régis Clouard

pgr2rg

Construction d'une carte de régions par application d'un graphe sur une carte de régions.

Synopsis

pgr2rg [*-m mask*] [*gr_in*|-] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pgr2rg** crée une carte de régions *rg_out* par sélection des régions de la carte de régions d'entrée *rg_in* correspondant à des sommets actifs du graphe d'entrée *gr_in*.

Les labels de sorties sont ceux de la carte de région d'entrée. Il peut donc y avoir des "trous" dans la liste des labels utilisée.

Entrées

- *gr_in*: un graphe.
- *rg_in*: une carte de régions.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit la carte de régions rout.pan avce les régions de rin.pan qui sont indexées par un noeud du graphe g.pan:

```
prg2gr g.pan rin.pan rout.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PGr2Rg( const Graph2d &gr_in, const Reg2d &rg_in, Reg2d &rg_out
);
```

Auteur: Régis Clouard

pgradient

Calcul du module et de la direction du gradient.

Synopsis

pgradient *halfsize* [*-m mask*] [*im_in*|-] [*im_out1*|-] [*im_out2*|-]

Description

L'opérateur **pgradient** calcule la derivée première de l'image d'entrée *im_in*. Le résultat est une image d'amplitude du gradient dans *im_out1* du même type que l'image *im_in* et une image de direction de ce gradient dans *im_out2* qui est de type Char.

La valeur d'amplitude du gradient en un point reflète la variation de niveau de gris observée dans l'image *im_in* en ce point. Plus cette valeur est élévée plus cette variation est forte. L'amplitude est obtenue par le maximum de la dérivée en x et en y (et en z en 3D).

La direction du gradient est orthogonale à la frontière qui passe au point considéré. Elle est obtenue par l'artang(dy/dx) suivie d'une discrétisation pour obtenir les valeurs en code de Freeman. L'image de direction est donc une image contenant des codes de Freeman [0..7] en 2D, [0..25] en 3D.

Les codes de Freeman:

```
  en 2D                 en 3D:
                          z-1:          z:              z+1:
 1 2 3                  2  3  4       10 11 12        19 20 21
 0   4                  1  0  5       9     22        18 13 14
 7 6 5                  8  7  6       25 24 23        17 16 15
```

Le calcul de la dérivée se fait par convolution de l'image avec un masque -1, 0, 1 dans toutes les directions. La valeur d'amplitude est prise que celle qui est maximale.

Pour les images couleur, c'est l'algorithme de Di Zenzo qui est utilisé. Il est basé sur la recherche des valeurs propres de la matrice:

```
  | p t |
  | t q |
où p=gxR*gxR+gxG*gxG+gxB*gxB
où q=gyRy*gyR+gyG*gyG+gyB*gyB
où t=gxR*gyR+gxG*gyG+gxB*gyB
```

Le module du gradient est donné par:

```
module=sqrt(lambda1 + lambda2)
avec lambda1=1/2 * (p+q + sqrt((p-q)*(p-q)-4*t*t))
     lambad2=1/2 * (p+q - sqrt((p-q)*(p-q)-4*tt))
```

et l'orientation est donnée par:

```
orientation = 1/2 *arctan (2*t / (p-q))
```

suivi d'une discrétisation selon le codage de Freeman.

Paramètres

- *halfsize* permet de spécifier la taille du masque de convolution.

Entrées

- *im_in*: une image.

Sorties

- *im_out1*: une image du même type que l'image im_in.
- *im_out2*: une image de Uchar.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Détection de contours basée sur le seuillage par hystérésis:

```
pgradient 1 tangram.pan i1.pan i2.pan
pbinarization 30 1e30 i1.pan i3.pan
pbinarization 60 1e30 i1.pan i4.pan
pgeodesicdilation 1 1 -1 i4.pan i3.pan i4.pan
surperimposition 0 tangram.pan i4.pan out.pan
```

Voir aussi

Détection de contours

Prototype C++

```
Errc PGradient( const Img2duc &im_in, Img2duc &im_out1, Img2duc
&im_out2, int halfsize );
```

Auteur: Régis Clouard

pgradientthreshold

Estimation du bruit dans une image d'amplitude du gradient.

Synopsis

pgradientthreshold *percent* [*-m mask*] [*im_in*|-]

Description

L'opérateur **pgradientthreshold** calcule la valeur de niveau de gris qui est supposée séparer les vrais contours des faux contours dus au bruit.

La valeur de seuil est calculée à partir de l'histogramme cumulé des valeurs d'amplitude du gradient. La valeur de seuil est la valeur d'amplitude maximale qui représente 1-*percent* valeurs de l'histogramme cumulé.

Cette valeur peut être récupérée par la commande **pstatus**.

Paramètres

- *percent* est un entier de l'intervalle [0..1] qui représente le nombre approximatif de points de contour dans l'image. Ce nombre est en général estimé autour de 0.2 (20%).

Entrées

- *im_in*: une image d'octets.

Résultat

Retourne la valeur d'amplitude du gradient.

Exemples

Détection de contours dans l'image tangram.pan:

```
pexponentialfiltering 0.7 tangram.pan i1.pan
pgradient 1 i1.pan i2.pan i3.pan
pnonmaximasuppression i2.pan i3.pan i4.pan
ppostthinning i4.pan i5.pan
pgradientthreshold 0.03 i2.pan
seuilhaut=‘pstatus‘
pbinarization $seuilhaut 1e30 i5.pan i6.pan
pgradientthreshold 0.2 i2.pan
seuilbas=‘pstatus‘
pbinarization $seuilbas 1e30 i5.pan  i7.pan
pgeodesicdilation 1 1 -1 i6.pan i7.pan out.pan
```

Voir aussi

Détection de contours

Prototype C++

```
Errc PGradientThreshold( const Img2duc &im_in, float percent );
```

Auteur: Régis Clouard

pgradneumann

Calcul du gradient d'une image par différences finies décentrées à droite avec conditions aux bords de Neumann.

Synopsis

pgradneumann [*-m mask*] [*im_in*|-] [*im_out1*|-] [*im_out2*|-]

Description

L'opérateur **pgradneumann** calcule la dérivée première de l'image d'entrée *im_in*. Le résultat est deux images de gris, où *im_out1* est la dérivée le long de l'axe x et *im_out2* est la dérivée le long de l'axe y:

```
im_out1(i,j) = im_in(i+1,j)-im_in(i,j),
im_out2(i,j) = im_in(i,j+1)-im_in(i,j), avec im_out1(n-1) = 0 et im_out2(n-1) = 0.
```

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out1*: une image de même type que im_in.
- *im_out2*: une image de même type que im_in.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Implémente le calcul du gradient et de la divergence avec les condition aux bords de Neumann, telle que façon que l'une est l'adjoint de l'autre, i.e. <grad x,u> = <-div u,x>. Le script vérifie l'identité de deux images.

```
protation 0 180 tangram.pan tangram1.pan
pgradneumann tangram.pan gim1_y.pan gim1_x.pan
pgradneumann tangram1.pan gim2_y.pan gim2_x.pan

# Compute < grad im1, grad im2>.
pmult gim1_y.pan gim2_y.pan | psumvalue - s1.pan
sumvaly=‘pstatus‘
pmult gim1_x.pan gim2_x.pan | psumvalue - s2.pan
sumvalx=‘pstatus‘
```

```
innerproduct1=‘echo "$sumvaly+$sumvalx" | bc -l‘

# Compute <-div grad im1,im2>.
pdivneumann gim1_y.pan gim1_x.pan | pmultcst -1 - divim1.pan
pim2sf tangram1.pan t.pan
pmult divim1.pan t.pan | psumvalue - /dev/null
innerproduct2=‘pstatus‘

echo $innerproduct1
echo $innerproduct2
```

Voir aussi

Edge detection, pdivneumann

Prototype C++

```
Errc Errc PGradNeumann( const Img2d<U> &im_in, Img2d<U> &im_out1,
Img2d<U> &im_out2 );
```

Auteur: Jalal Fadili

pgraphbasedsegmentation

Segmentation d'images couleur par l'analyse des frontières des régions.

Synopsis

pgraphbasedsegmentation *sigma k minimum-region-area* [*-m mask*] [*im_in*|*-*] [*rg_out*|*-*]

Description

L'opérateur **pgraphbasedsegmentation** exploite la frontière entre les régions pour segmenter des images couleur. Il adopte un représentation par graphe de la région et utilise l'homogénéité de l'intensité inter et intra-régions pour déterminer les frontières. La différence d'intensité à l'intérieur d'une région est définie comme étant la plus grande masse des bords de l'arbre couvrant minimum construit à partir de cette région, et la différence d'intensité entre région est définie comme étant le poids minimum de la frontière qui relie ces deux régions.

Le résultat est la carte de la région *rg_out*.

Paramètres

- *sigma* est utilisé pour lisser l'image d'entrée avant de la segmenter.
- *k* est la valeur de la fonction de seuillage. Elle contrôle le degré de finesse de la segmentation.
- *minimum-region-area* spécifie la taille minimale (en pixel) des régions de sortie. C'est un entier supérieur à 0 donné en pixels.

Entrées

- *im_out*: une image 2D.

Sorties

- *im_out*: une carte de régions.

Résultat

Retourne le nombre de régions formées ou FAILURE.

Exemples

Segmente les pièces de tangram:

```
pgraphbasedsegmentation 0.5 500 20 examples/tangram.pan a.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PGraphBasedSegmentation( const Imc2duc &ims, Reg2d &rgd, float sigma, float threshold, int minimumRegionArea );
```

Référence

Pedro F. Felzenszwalb and Daniel P. Huttenlocher, "Efficient Graph-Based Image Segmentation", International Journal of Computer Vision, 59(2) September 2004.

pgraphneighbours

Valuation des sommets d'un graphe avec le nombre de sommets voisins.

Synopsis

pgraphneighbours [*gr_in*|-] [*gr_out*|-]

Description

L'opérateur **pgraphneighbours** permet d'attribuer à chaque sommet de *gr_out* le nombre de voisins qui lui sont attachés.

Entrées

- *gr_in*: un graphe.

Sorties

- *gr_out*: un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pgraphneighbors g1.pan g2.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PGraphNeighbours( const Graph &gr_in, Graph &gr_out );
```

Auteur: François Angot

pgraphpruning

Suppression des arcs nuls et des sommets isolés.

Synopsis

pgraphpruning [*-m mask*] [*gr_in*|-] [*gr_out*|-]

Description

L'opérateur **pgraphpruning** permet de couper les arcs d'un sommet de valeur nulle avec tous ses voisins. De plus, si un sommet n'a pas de voisin, son champ *value* est mis à 0.

Entrées

- *gr_in*: un graphe.

Sorties

- *gr_out*: un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pgraphpruning g1.pan g2.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PGraphPruning( const Graph &gr_in, Graph &gr_out );
```

Auteur: François Angot

pgraphvisu

Visualisation des valeurs des sommets et des arcs d'un graphe.

Synopsis

pgraphvisu [*-m mask*] [*gr_in*|-] [*im_out*|-]

Description

L'opérateur **pgraphvisu** permet de visualiser dans une image les valeurs des sommets et des arcs du graphe d'entrée *gr_in*.

L'image de sortie est donc de type float.

Entrées

- *gr_in*: un graphe 2D ou 3D.

Sorties

- *im_out* : une image de type float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pvisugraph g.pan out.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PGrapheVisu( const Graph2d &gr_in, Img2dsf &im_out );
```

Auteur: Régis Clouard

pgray2bw

Conversion d'une image de niveaux de gris en une image binaire équivalente.

Synopsis

pgray2bw [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pgray2bw** est un moyen de convertir une image de niveaux de gris en image noir et blanc.

Chaque pixel de l'image d'entrée *im_in* est converti en valeur booléenne. Pour cela, ll faut utiliser une matrice de probabilité indiquant une valeur de niveau de gris minimum pour que le point soit mis 255. Sinon il est conservé à 0.

La matrice 16x16 utilisé par l'opérateur est :

```
{128,   32,    160,   8,     136,   40,    168,   2,     130,   34,    162,   10,    138,   42,    170,   192},
{64,    224,   96,    200,   72,    232,   104,   194,   66,    226,   98,    202,   74,    234,   106,   48},
{176,   16,    144,   56,    184,   24,    152,   50,    178,   18,    146,   58,    186,   26,    154,   240},
{112,   208,   80,    248,   120,   216,   88,    242,   114,   210,   82,    250,   122,   218,   90,    12},
{140,   44,    172,   4,     132,   36,    164,   14,    142,   46,    174,   6,     134,   38,    166,   204},
{76,    236,   108,   196,   68,    228,   100,   206,   78,    238,   110,   198,   70,    230,   102,   60},
{188,   28,    156,   52,    180,   20,    148,   62,    190,   30,    158,   54,    182,   22,    150,   252},
{124,   220,   92,    244,   116,   212,   84,    254,   126,   222,   94,    246,   118,   214,   86,    3},
{131,   35,    163,   11,    139,   43,    171,   1,     129,   33,    161,   9,     137,   41,    169,   195},
{67,    227,   99,    203,   75,    235,   107,   193,   65,    225,   97,    201,   73,    233,   105,   51},
{179,   19,    147,   59,    187,   27,    155,   49,    177,   17,    145,   57,    185,   25,    153,   243},
{115,   211,   83,    251,   123,   219,   91,    241,   113,   209,   81,    249,   121,   217,   89,    15},
{143,   47,    175,   7,     135,   39,    167,   13,    141,   45,    173,   5,     133,   37,    165,   207},
{79,    239,   111,   199,   71,    231,   103,   205,   77,    237,   109,   197,   69,    229,   101,   63},
{191,   31,    159,   55,    183,   23,    151,   61,    189,   29,    157,   53,    181,   21,    149,   254},
{254,   127,   223,   95,    247,   119,   215,   87,    253,   125,   221,   93,    245,   117,   213,   85}};
```

Pour chaque pixel de l'image d'entrée:

```
if (im_in[p] >= matrix[p.y%16][p.x%16])
   imd[p]=255;
else
   imd[p]=0;
```

Entrées

- *im_in*: une image 2D d'octets (Img2duc).

Sorties

- *im_out*: une image 2D d'octets (Img2duc).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Transforme l'image 'tangram.pan' en image noir et blanc.

```
pgray2bw tangram.pan a.pan
```

Voir aussi

Color

Prototype C++

```
Errc PGray2BW( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

pgray2falsecolor

Conversion d’une image de niveaux de gris en une image avce fausses couleurs.

Synopsis

pgray2falsecolor *lut* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L’opérateur **pgray2falsecolor** est un moyen de convertir une image de niveaux de gris en image noir et blanc.

Chaque pixel de l’image d’entrée *im_in* est converti en une valeur de rouge, de vert et de bleu selon une table de couleur donnée.

Pour chaque pixel de l’image d’entrée:

```
imd.R[p]=lut[0][ims[p]
imd.V[p]=lut[1][ims[p]
imd.B[p]=lut[2][ims[p]
```

Paramètres

- *lut* spécifie le numéro de la lut choisie parmi:
  - 0: "arc en ciel"
  - 1: prédéfinie sur 15 couleurs différentes.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *im_out*: une image couleur.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Transforme l’image ’tangram.pan’ en image utilisant la table des couleurs "arc en ciel".

```
pgray2falsecolor 0 tangram.pan a.pan
```

Voir aussi

Color

Prototype C++

```
Errc PGray2FalseColor( const Img2duc &im_in, Imc2duc &im_out, int
lutId );
```

Auteur: Régis Clouard

pgraylevel2depth

Construction d'une image de reliefs 3D à partir d'une image 2D.

Synopsis

pgraylevel2depth *depthmax* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pgraylevel2depth** construit une image 3D à partir d'une image 2D, où les valeurs de niveaux de gris sont converties en profondeur. Par exemple, le niveau de gris 127 à la coordonnée (x,y) est recopié sur les 127 plans de l'image résultat à la même coordonnée.

Le paramètre *depthmax* définit la profondeur de l'image de sortie *im_out*. Toutes les profondeurs sont normalisées à partir de cette profondeur maximale.

L'mage 3D est construite à partir du dernier plan. Cela signifie que les objets clairs occupent les premiers plans et les objets sombres les derniers.

L'algorithme est le suivant:

```
for (y=0; y< normalize(im_in[x],depthmax); y++)
        im_out[y][x] = normalize(im_in[x],depth);
```

Paramètres

- *depthmax* définit la profondeur maximale de l'image de sortie *im_out*.

Entrées

- *im_in*: une image 2D de niveaux de gris.

Sorties

- *im_out*: a 3D Uchar image (img3duc).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit l’image 3D out.pan à partir de l’image 2D tangram.pan:

```
pgraylevel2depth 50 tangram.pan a.pan
```

Voir aussi

Utilitaire, pdepth2graylevel

Prototype C++

```
Errc PGraylevel2Depth( const Img2duc &im_in, Img3duc &im_out, long depthmax );
```

Auteur: Jean-Marie Janik

pharris

Détection de points d'intérêt selon l'algorithme de Harris-Stephens.

Synopsis

pharris *sigma kappa* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pharris** permet de détecter les points d'intérêt dans l'image d'entrée *im_in*. Les points d'intérêt sont soit des coins en L, des jonctions en T, des jonctions en Y ou des points de forte variation de texture. Ils correspondent à des doubles discontinuités de la fonction d'intensité provoquées par des discontinuités de la fonction réflectance ou de profondeur.

Le principe de l'algorithme est de calculer une matrice de covariance C(x,y):

```
    | Ixx Ixy |
C=|         |
    | Ixy Iyy |
```

où Ixx, Iyy et Ixy sont respectivement la somme des valeurs carrées de gradient en X, en Y et en X et Y dans un voisinage (2*halsize+1)x(2*halfsize+1) autour du pixel (x,y).
Si la plus petite valeur propre de cette matrice au point p est positive alors ce point est considéré comme un point d'intérêt.

Pour éviter de calculer les valeurs propres, Harris propose de calculer la fonction de réponse R(x,y) pour chaque pixel par:

```
R=Ixx*Iyy-Ixy*Ixy - kappa * (Ixx+Iyy) * (Ixx+Iyy)
```

puis de rechercher les maxima locaux de la fonction R.

Pratiquement, pour chaque pixel q dans le voisinage de p, le gradient est [Ix,Iy], C est la matrice de covariance de tous les vecteurs gradients dans le voisinage de p. Les valeurs propres representent le grand axe et le petit axe de l'ellipse approximant la distribution des vecteurs gradient.

L'image de sortie *im_out* est une image de float qui code pour chaque pixel, la force de la réponse.

Paramètres

- *sigma* est l'éart-type de la gaussienne, et donne aussi la taille de la zone de recherche du maximum local (largeur=6*sigma). Une valeur typique appartient à [1..3].
- *kappa* est un facteur de pondération maximum pour que R soit positif. La valeur estimée par Harris est de 0.04.

Entrées

- *im_in*: une image d'intensité 2D.

Sorties

- *im_dest*: une image de float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Extrait les coins de l'image tangram.pan et superimposes les coins dans l'image initiale.

```
pharris 2 0.04 tangram.pan a.pan
pbinarization 1e4 1e30 a.pan b.pan
padd b.pan tangram.pan out.pan
```

Voir aussi

Points d'intérêt

Prototype C++

```
Errc PHarris( const Img2duc &im_in, Img2dsf &im_out, float sigma,
float kappa );
```

Auteur: Régis Clouard

phermiterescale

Retaille d'une image par l'algorithme de Hermite.

Synopsis

phermiterescale *zoomx zoomy zoomyz* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **phermetirescale** utilise un noyau de convolution pour interpoler les valeurs des pixels de l'image d'entrée *im_in* afin de calculer les valeurs des pixels de l'image de sortie *im_out*.
L'interpolation consiste à pondérer l'influence des pixels d'entrée. Les poids sont dépendants de la position du pixel de sortie et sont donnés par l'algorithme de Bell:

```
       | (2*x - 3)*x*x + 1 if -1<x<1
H(x) = | 0 otherwise
```

Par exemple, si l'image est zoomée de 3, alors chaque pixel résultat est :

```
for i in [-2, 2]
  for j in [-2, 2]
      im_out[p.y][p.x] += H(i*scalex)*H(j*scaley)*im_in[p.y*scaley+j][p.x*scalex+i]
```

Pour rezoomer une carte de région ou un graphe, il faut utiliser l'opérateur prescale.

Paramètres

- *rescalex*, *rescaley*, *rescalez* sont des réels positifs correspondant aux facteurs de retaille.
  Si les rescales sont > 1 alors il s'agit d'un agrandissement, s'ils sont < 1 alors il s'agit d'une réduction.
  rescalez est ignoré pour le cas des images 2D mais doit être donné.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Agrandissement de l’image d’un facteur 2 :

```
phermiterescale 2 2 0 tangram.pan a.pan
```

Réduction de l’image d’un facteur 2 :

```
phermiterescale 0.5 0.5 0 tangram.pan a.pan
```

See also

Transformation, plinearrescale, pbicubicrescale, planczosrescale, pmitchellrescale, pquadraticbsplinerescale, prescale

Prototype C++

```
Errc PHermiteRescale( const Img2duc &im_in, Img2duc &im_out, const float zoomy, const float zoomx );
```

Author: Régis Clouard

phistogram

Création de l'histogramme d'une image.

Synopsis

phistogram [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **histogramme** génère l'histogramme de l'image *im_in*. Un histogramme est un signal dont l' numéro *i* indique le nombre d'apparition de la valeur *i* dans l'image source.

Pour les images signées, l'histogramme est décalé de telle sorte que le premier élément indique le nombre d'apparition de MIN dans l'image source et le dernier MAX-MIN.

Le type de l'image de sortie *im_out* est une image 1D de Long.

Entrées

- *im_in*: une image de niveaux de girs.

Sorties

- *im_out*: une image 1D de Long.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit l'histogramme de l'image tangram.pan:

```
phistogram tangram.pan a.pan
pplot1d 256 256 0 0 0 a.pan b.pan
pvisu b.pan
```

Voir aussi

Caractérisation image

Prototype C++

```
Errc PHistogram( const Img2duc &im_in1, Img1dus &im_in2 );
```

Auteur: Alexandre Duret-Lutz

phistogramequalization

Rehaussement de contraste par égalisation d'histogramme.

Synopsis

phistogramequalization *min max* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **phistogramequalization** effectue une transformation des niveaux de gris en étalant les niveaux de gris sur toute la plage de valeurs définie par les bornes *min* et *max*. En conséquence, l'équalisation d'histogramme étale les niveaux de gris où l'information est dense et compresse les niveaux de gris où l'information est éparse.
La transformation opère moins bien quand l'image d'entrée est sombre.

Les nouvelles valeurs de borne de l'image de sortie sont données par les paramètres *min* et *max*.

L'équalisation s'effectue en 3 étapes :

1. Calcul de l'histogramme cumulé *im_in*;
2. Normalisation de l'histogramme entre [0..1];
3. Construction de l'image de sortie *im_out* avec pour chaque pixel 'p':

```
im_out[p] = HC[im_in[p]]*(max-min);
```

Pour les images couleur et multispectrale, la transformation utilise l'approche marginale : l'opérateur est appliqué sur chaque bande individuellement.

Paramètres

- *min* et *max* spécifie les bornes pour les valeurs des pixels de l'image de sortie. Les valeurs possibles dépendent du type de l'image d'entrée.
  Note: si *min* > *max* alors min et max prennent la valeur du minimum et du maximum possible.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image avec les mêmes propriétés que l'image d'entrée *im_in*.

Résultat

Retourne SUCCESS ou FAILURE en cas de mauvais paramétrage.

Exemples

- Equalisation de l'histogramme avec les nouvelles bornes [0,255] :

```
phistogramequalization 20 200 tangram.pan a.pan
```

- Equalisation de l'histogramme en gardant les mêmes bornes :

```
phistogramequalization 0 -1 tangram.pan a.pan
```

Voir aussi

Transformation de la LUT, phistogramspecification

Prototype C++

```
Errc PHistogramEqualization( const Img2duc &im_in, Img2duc &im_out,
float min, float max );
```

Auteur: Régis Clouard

phistogramspecification

Rehaussement de contraste par spécification d'histogramme.

Synopsis

phistogramspecification [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **phistogramspecification** effectue une transformation des niveaux de gris de l'image d'entrée *im_in1* en utilisant une forme d'histogramme particulière donnée par l'image de référence *im_in2*.

Cet opérateur peut être utilisé pour améliorer une liste d'images d'une même scène. La première étape est d'améliorer l'histogramme d'une image de la liste à la main, et la deuxième étape consiste à appliquer le même histogramme sur toutes les images de la liste.

La spécification d'histogramme est basé sur l'algorithme suivant:

1. calcule l'histogramme cumulé normalisé hc1 de l'image d'entrée *im_in1*;
2. calcule l'histogramme cumulé normalisé hc2 de l'image de référence *im_in2*;
3. pour chaque pixel 'p' de l'image d'entrée :
    1. s=hc1[im_in1[p]];
    2. Search for i such as hc2[i]=s;
    3. im_out[p]=i.

Pour les images couleur et multispectrale, la transformation utilise l'approche marginale : l'opérateur est appliqué sur chaque bande individuellement.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image du même type que l'image d'entrée *im_in*.

Résultat

Retourne SUCCESS ou FAILURE en cas d'incompatibilité entre les images d'entrée.

Exemples

Applique la transformation de l'histogramme de l'image tangram.pan par une transformation logarithmique à l'image tangram.pan elle-même :

```
plogtransform 0 0 255 tangram.pan reference.pan
phistogramspecification tangram.pan reference.pan a.pan
```

Voir aussi

Transformation de la LUT, phistogramequalization

Prototype C++

```
Errc PHistogramSpecification( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc &im_out );
```

Auteur: Régis Clouard

phistomerging

Fusion prioritaire de régions selon la corrélation d'histogramme.

Synopsis

phistomerging *nb_fusion seuil* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

L'opérateur **phistomerging** permet de fusionner les régions de la carte de régions *rg_in* selon le critère de la corrélation d'histogrammes.

La notion de voisinage entre les régions est détenue par le graphe *gr_in*.

Le principe de l'algorithme est le suivant:
Pour chaque région de la carte de régions *im_in*, on calcule le coefficient de corrélation entre les histogrammes de la région et de ses voisines.
Si le coefficient est supérieur au *seuil* donnée. paramètre, les régions sont fusionnées.
On utilise ici l'algorithme de croissance prioritaire qui consiste à fusionner à chaque fois les 2 régions dont la corrélation est la plus élévée.

La corrélation d'histogrammes entre 2 régions est calculée par:

```
correlation(R1,R2) = H1.H2 / (norme(H1).norme(H2))
H1.H2 = produit scalaire
norme(Hi) = norme euclidenne de l'histogramme Hi
```

Plus le coefficient est grand plus les 2 histogrammes sont ressemblants.

Cet opérateur fonctionne mal avec des petites régions parce que les histogrammes ne sont pas assez significatifs.

Paramètres

- *nb_fusion* permet de spécifier le nombre de fusions à effectuer (la valeur -1 signifie d'ignorer ce paramètre et d'exécuter l'algorithme tant qu'il y a des fusions possibles).
- *seuil* permet de spécifier la tolérance par rapport au critère de corrélation d'histogramme entre deux régions. Les valeurs appartiennent à l'intervalle [0..1], où 1 correspond à 2 histogrammes strictement égaux. On prendra généralement des valeurs proches de 1 (e.g., 0.7).

Entrées

- *rg_in*: une carte de régions.
- *gr_in*: un graphe.
- *im_in*: une image.

Sorties

- *rg_out*: une carte de régions.
- *gr_out*: un graphe.

Résultat

Retourne le nombre de fusions effectuées.

Exemples

Fusionne les régions issue d'une partition de tangram.pan :

```
puniformityquadtree 0.9 tangram.pan a.pan
prg2gr a.pan b.pan
phistomerging -1 0.94 a.pan b.pan tangram.pan c.pan d.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PHistoMerging( const Reg2d &rg_in, const Graph2d &gr_in, const
Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, Long nb_fusion,
Uchar seuil );
```

Auteur: Laurent Quesnel

phistothresholding

Multi-thresholding using histogram thresholding.

Synopsis

phistothresholding *length* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **phistothresholding** permet de multiseuiller l'image initiale *im_in* par analyse de l'histogramme des niveaux de gris.
Chaque maximum de l'histogramme de l'image d'entrée *im_in* correspond à une classe dans l'image de sortie *im_out*. L'histogramme de l'image est préalablement lissé par une lissage exponentiel pour éliminer les faux maxima.

Une ligne de partage des eaux est ensuite appliquée à l'histogramme de manière à le segmenter.

La recherche des maxima locaux de P(n) est faite sur une plage maximale de *length* niveaux de gris de part et d'autre du niveau de gris n.

L'image de sortie *im_out* est construite avec les seuils détectés, telle que:

`im_out[y][x]=seuil[k] si seuil[k-1]<im_out[y][x]<=seuil[k].`

Le dernier seuil est égal à la valeur maximale 255.

Paramètres

- *length* définit la plage de recherche des maxima de l'histogramme. Plus ce paramètre est grand, moins il y a de maxima régionaux et donc moins de classes en sortie.

Entrées

- *im_in*: une image de niveaux de gris d'octets (Img2duc, Img3duc).

Sorties

- *im_out*: une image du même type que l'entrée.

Résultat

Retourne le nombre de classes détectées.

Exemples

Segmente les pièces de tangram:

```
phistothresholding 10 tangram.pan out.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PHistoThresholding( const Img2duc &im_in, Img2dsl &im_out, int
length );
```

Auteur: Olivier Lezoray

phitormiss

Transformation de type tout ou rien.

Synopsis

phitormiss [*-m mask*] [*im_se1*|-] [*im_se2*|-] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **phitormiss** permet de trouver la position d'une forme parmi un ensemble de formes. La forme est définie par deux éléments structurants : im_se1 est un élément structurant qui spécifie les parties de l'objet, et im_se2 spécifie les parties du fond. La transformation peut être résumée par la question: "Est-ce que im_se1 détecte l'objet tandis que im_se2 détecte le fond de l'objet ?"

```
[UHMT(f)](x) = [erod_se1(f)](x) - [dil_se2](x) si [erod_se1(f)](x) > [dil_se2](x)
               0 sinon
```

Entrées

- *im_se1*: une image d'octets.
- *im_se2*: une image d'octets.
- *im_in*: une image.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Détecte des bouts de lignes droites dans une image de gradient :

```
cat > se1.txt << EOF
255 0 3
255 1 3
255 2 3
255 3 3
255 4 3
EOF
cat > se2.txt<< EOF
255 2 0
255 2 1
255 2 5
```

```
255 2 6
EOF
ptxt2pan 0 5 7 0 se1.txt se1.pan
ptxt2pan 0 5 7 0 se2.txt se2.pan
pgradient 1 examples/tangram.pan a.pan b.pan
phitormiss se1.pan se2.pan a.pan c.pan
```

Voir aussi

Morphologie,

Prototype C++

```
Errc PHitOrMiss( const Img2duc &im_in, const Img2duc &im_se1, const
Img2duc &im_se2, Img2duc &im_out, int size );
```

Auteur: Régis Clouard

pholeselection

Sélection des trous dans les régions.

Synopsis

pholeselection *connexity* [-m mask] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur retourne une carte de régions *rg_out* localisant les trous dans les régions de la carte d'entrée *rg_in*.

Un trou est une région du fond (ie, label=0) qui n'a qu'une seule région voisine.

Paramètres

- *connexity* spécifie le type de connexité pour les pixels d'un trou (4 ou 8 pour le 2D; 6 ou 26 pour le 3D).

Entrées

- *rg_in*: une carte de régions.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de trous sélectionnés.

Exemples

Sélection des trous dans les pièces de tangram :

```
pbinarization 97 255 examples/tangram.pan
plabeling 8 a.pan b.pan
pholeselection 8 b.pan c.pan
```

Voir aussi

Region

Prototype C++

```
Errc PHoleSelection( const Img2duc &rg_in, Img2duc &rg_out, int
connexity );
```

Auteur: Régis Clouard

phomotopicskeletonization

Squelettisation homotopique d'objets binaires.

Synopsis

phomotopicskeletonization *connexity* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **squelettisation_homotopique** permet d'obtenir le noyau homotopique d'une image 3D. L'algorithme repose sur une suppression séquentielle des points simples. un point simple est point dont la suppression préserve la topologie de l'image.

Paramètres

- *connexity* est une valeur de connexité qui vaut 6 ou 26, selon que les objets sont considérés comme 6-connexes (et le fond 26-connexe) ou bien 26-connexes (et le fond 6-connexe).

Entrées

- *im_in*: une image binaire 3D.

Entrées

- *im_out*: une image binaire 3D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Construit le squelette homotopique d'un parallélepipède (il sera à un point) :

```
pshapedesign 256 256 128 13 120 80 a.pan
phomotopicskeletonization 26 a.pan out.pan
```

Voir aussi

Morphologie

Prototype C++

```
Errc PHomotopicSkeletonization( const Img2duc &im_in, Img2duc
&im_out, int connexity );
```

Auteur: Sébastien Fourey

phoughlines

Détection et localisation des segments de droite dans une image de contours par la transformée de Hough.

Synopsis

phoughlines *lines minangle maxangle linethickness* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **phoughlines** permet de détecter des droites sur une image de contours. L'image d'entrée *im_in* doit être une image de contours pour être exploitable par l'opérateur. L'image de sortie *im_out* contient les droites correspondant aux segments de droites détectés dans l'image initiale *im_in*.

Principe de la transformée de Hough :
La méthode de Hough permet de reconnaître des équations géométriques dans une image. Pour cela, on utilise un accumulateur qu'on appelle l'"espace de Hough" qui est un tableau qui a autant de dimensions que l'équation de la forme géométrique recherchée a de paramètres.

Pour détecter les droites, on considère l'équation générale des droites :

```
rho = x*cos(theta) + y*sin(theta).
```

où rho est la perpendiculaire à l'origine et theta l'angle avec la normale.

On parcourt alors chaque point du contour de l'image et on recherche pour ce point toutes les équations de droites auxquelles peut appartenir ce point. Les paramètres de l'équation donnent une coordonnée dans l'espace de Hough que l'on incrémente. Ainsi, les valeurs de l'accumulateur définissent un nombre de vote pour chaque coordonnée. Puis on recherche dans l'accumulateur les paramètres qui ont réuni le maximum de vote, c'est eux qui donneront les paramètres de l'équation qui ont été la plus vérifiée. Il ne reste donc plus qu'à dessiner dans l'image de sortie la forme correspondant à l'équation trouvée.

L'algorithme revient à :

- initialiser l'accumulateur
- incrémenter la case accumulateur(r,o) pour chaque (y,x) vérifiant rho=x*cos(theta)+y*sin(theta)

  Pour chacune des *lines* droites faire:
  - rechercher le maximum de l'accumulateur
  - dessiner la droite correspondant à ce maximum
  - supprimer le maximum dans l'accumulateur.

Pour contourner le problème des "lignes fantômes" dues à la discrétisation des points de contour, la suppression du maximum se fait en supprimant tous les points de contour de la droite détecté dans l'image initiale ainsi que ceux des lignes qui sont à distance inférieure à *linetickness*, puis par recalcul de l'accumulateur (en fait une version améliorée de cette solution).

Paramètres

- *lines* est le nombre de droite que l'on veut trouver en sortie.
- *minangle* et *minangle* contrôlent l'angle de recherche des droites. Seules les lignes comprises entre les deux angles sont retenus. Les angles sont mesurés en degré et ont une valeur entre -360 + 360. Pour sélectionner toutes les lignes il faut utiliser minagle=0 et maxangle=180.
- *linethickness* permet de ne sélectionner que des lignes qui sont à distance minimale de *linethickness*.

Entrées

- *im_in*: une image de type Uchar (Img2duc).

Sorties

- *im_out*: une image de Uchar.

Résultat

Retourne le nombre de lignes détectées.

Exemples

Extrait les lignes droites à partir d'un ensemble de contours obtenus par une simple détection de contours :

```
psobel tangram.pan b.pan
pbinarization 45 1e30  b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
phoughlines 20 0 180 2 e.pan f.pan
pimg2imc 0 f.pan c.pan c.pan out.pan
```

Voir aussi

Contour

Prototype C++

```
Errc PhoughLines( const Img2duc &im_in, Img2duc &im_out, int lines,
int minangle, int maxangle, int thickness );
```

Auteur: Laurent Quesnel

phsi2rgb

Changement d'espace couleur de HSI vers RGB.

Synopsis

phsi2rgb [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **phsi2rgb** permet de changer l'espace couleur HSI (Teinte, Saturation, Intensité) vers vers l'espace RGB (Rouge, Vert, Bleu).

Entrées

- *im_in*: une image couleur hsi.

Sorties

- *im_out*: une image couleur RGB.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image parrot.pan de rgb en hsi et réciroquement.

```
prgb2hsi parrot.pan a.pan
phsi2rgb a.pan b.pan
```

Voir aussi

Color, prgb2hsi

Prototype C++

```
Errc PHSI2RGB( const Imc2dsf &im_in, Imc2dsf &im_out );
```

Auteur: Olivier Lezoray

phsl2rgb

Changement d'espace couleur de HSL vers RGB.

Synopsis

phsl2rgb [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **phsl2rgb** permet de changer l'espace couleur HSL (Teinte, Saturation, Luminance) vers vers l'espace RGB (Rouge, Vert, Bleu).

La teinte (Hue) est la qualité de couleur correspondant à sa position dans le spectre: rouge, orange, jaune, vert, cyan, bleu, magenta. Elle s'exprime en degré [0,360].

La saturation est l'intensité d'une couleur. Elle s'exprime par une valeur d'intensité de pourcentage de l'intervalle [0..100]. A 0% de saturation, une couleur apparaît blanche, à 100% de saturation, une couleur atteint son degré de plus intense.

La luminosité (Lightness) est la quantité de blanc et de noir contenue dans une couleur. Elle s'exprime par une valeur d'intensité de l'intervalle [0,255].

La conversion utilise la transformation suivante :

```
q= | l * (1+s),  si l <  1/2
   | l+s - (l*s) si l >= 1/2

p = 2 * l - q

t_k = t / 360

t_R = t_k+1/3
t_V = t_k
t_B = t_k-1/3

Pour chaque C dans {R,V,B}
  si t_C < 0 : t_C = t_C + 1.0
  si t_C > 1 : t_C = t_C - 1.0

      | p+((q-p)*6*t_C)        si t_C <1/6
      | q                     si 1/6 <= T_C <1/2
  C = | p+ ((q-p)*6*(2/3-T_C)) si 1/2 <= T_C < 2/3
      | p                     sinon
```

Entrées

- *im_in*: une image couleur hsl.

Sorties

- *im_out*: une image couleur RGB.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image parrot.pan de rgb en hsl et réciroquement.

```
prgb2hsl parrot.pan a.pan
phsl2rgb a.pan b.pan
```

Voir aussi

Color, prgb2hsl

Prototype C++

```
Errc PHSL2RGB( const Imc2dsf &im_in, Imc2dsf &im_out );
```

Auteur: Régis Clouard

phsv2rgb

Changement d'espace couleur de HSV vers RGB.

Synopsis

phsv2rgb [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **phsv2rgb** permet de changer l'espace couleur HSV (Teinte, Saturation, Value) vers vers l'espace RGB (Rouge, Vert, Bleu).

La teinte (Hue) est la qualité de couleur correspondant à sa position dans le spectre: rouge, orange, jaune, vert, cyan, bleu, magenta. Elle s'exprime en degré [0,360].

La saturation est l'intensité d'une couleur. Elle s'exprime par une valeur d'intensité de pourcentage de l'intervalle [0..100]. A 0% de saturation, une couleur apparaît blanche, à 100% de saturation, une couleur atteint son degré de plus intense.

La valeur est la plus forte composante couleur. Elle s'exprime par une valeur d'intensité de l'intervalle [0,255].

La conversion utilise la transformation suivante :

```
if (S == 0) {
    R = G = B = V
else
    H /= 60
    S /= 100

    w = | H |
    f = H - w;
    p = V * (1 - S);
    q = V * (1 - S * f);
    t = V * (1 - S * (1 - f));

   | R = V, G = t, B = p   if w = 0
   | R = q, G = V, B = p   if w = 1
   | R = p, G = V, B = t   if w = 2
   | R = p, G = q, B = V   if w = 3
   | R = t, G = p, B = V   if w = 4
   | R = V, G = p, B = q   if w = 5
```

Entrées

- *im_in*: une image couleur hsv.

Sorties

- *im_out*: une image couleur RGB.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image parrot.pan de rgb en hsv et réciroquement.

```
prgb2hsv parrot.pan a.pan
phsv2rgb a.pan b.pan
```

Voir aussi

Color, prgb2hsv

Prototype C++

```
Errc PHSV2RGB( const Imc2dsf &im_in, Imc2dsf &im_out );
```

Auteur: Régis Clouard

pidwt

Reconstruction d'une image décomposée en ondelettes dyadiques biorthogonales.

Synopsis

pidwt *scale* [*im_in*|-] [*col_in*|-] [*im_out*|-]

Description

L'opérateur **idwt** calcule l'image selon l'algorithme pyramidal complémentaire de celui utilisé par pdwt.

Les coefficients du filtre utilisé se trouvent dans la collection *im_in* créée à partir de l'opérateur pqmf.

Paramètres

- *scale* indique le nombre de niveaux de résolution sur lesquels a été décomposée l'image.

Entrées

- *im_in*: une image 2D de Float.
- *col_in*: une collection contenant les coefficients du filtre.

Sorties

- *im_out*: une image 2D de float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit une image synthétique avec un carré pour illustrer le phénomène de Gibbs an analyse par ondelettes:

```
pshapedesign 256 256 0 2 150 150 a.pan
pqmf daubechies 4 b.pan
pdwt 1 a.pan b.pan c.pan
psplitimage c.pan d1.pan d2.pan d3.pan d4.pan
pthresholding 20 400 d2.pan e2.pan
pthresholding 20 400 d3.pan e3.pan
pthresholding 20 400 d4.pan e4.pan
pmergeimages d1.pan e2.pan e3.pan e4.pan f.pan
pidwt 1 f.pan b.pan out.pan
```

Voir aussi

Domaine Fréquentiel, dwt, pqmf

Prototype C++

```
Errc PIdwt( const Img2dsf &ims, const Collection &c, Img2dsf &imd,
int scale=1 );
```

Auteur: Ludovic Soltys

pifft

Transformée de Fourier Rapide Inverse d'une image.

Synopsis

pifft [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out1*|-] [*im_out2*|-]

Description

L'opérateur **pifft** permet de calculer la transformée de Fourier inverse d'une image complexe. La transformée de Fourier d'une image est obtenue par la commande **pfft**.

Les images d'entrée sont de type réel:

- *im_in1* est la partie réelle de l'image fréquentielle.
- *im_in2* est la partie imaginaire de l'image fréquentielle.

Les images de sortie :

- *im_out1* est la partie réelle de l'image spatiale.
- *im_out2* est la partie imaginaire de l'image spatiale.

La transformée de Fourier Inverse permet de passer d'une représentation de l'image dans le domaine fréquentiel à une représentation dans le domaine spatial.

Entrées

- *im_in1*: une image de réels (partie réelle).
- *im_in2*: une image de réels (partie imaginaire).

Sorties

- *im_out1*: une image de réels (la partie réelle).
- *im_out2*: une image de réels (la partie imaginaire).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit une image de carré du domaine spatial dans le domaine fréquentiel et réciproquement :

```
pshapedesign 256 256 0 2 20 0 square.pan
pshapedesign 256 256 0 0  0 0 empty.pan
pfft square.pan empty.pan real.pan imaginary.pan
pmodulus real.pan imaginary.pan modulus.pan
pphase real.pan imaginary.pan phase.pan
pifft real.pan imaginary.pan square1.pan empty1.pan
plineartransform 0 0 255 square1.pan square2.pan
pim2uc square2.pan newsquare.pan
```

Voir aussi

Domaine Fréquentiel, pfft

Prototype C++

```
Errc IFFT( const Img2duc &im_in1, const Img2duc &im_in2, Img2dsf
&im_out1, Img2dsf &im_out2 );
```

Auteur: Herissay & Berthet

pim2array

Conversion d'une image en un vecteur dans une collection.

Synopsis

pim2array *name* [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pim2array** crée une collection *col_out* contenant un vecteur (tableau) *name* dans lequel les cases correspondent aux pixels de *im_in*.

Dans le cas d'une image couleur, trois tableaux: *name.1*, *name.2* et *name.3* correspondant respectivement aux canaux R, V et B sont créés.

Paramètres

- *name* est le nom du vecteur de pixels dans la collection.

Entrées

- *im_in*: une image.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image tangram.pan en vecteur foo dans la collection col.pan:

```
pim2array foo tangram.pan col.pan
```

Voir aussi

Coercition, parray2im

Prototype C++

```
Errc PIm2Array( const Img2duc &im_in, Collection &cd,std::string
name );
```

Auteur: Alexandre Duret-Lutz

pim2d23d

Construction d'une image ou d'une carte de régions 3D à partir d'une image ou d'une carte de régions 2D.

Synopsis

pim2d23d [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pim2d23d** construit une image 3D à un plan avec une image 2D.

Si l'entrée est une carte de régions 2D alors la sortie sera une carte de région 3D.

L'opérateur psetslice peut être utilisé pour insérer une image 2D dans une image 3D.

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: une image 3D à 1 plan.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image 2D tangram.pan en image 3D:

```
pim2d23d tangram.pan a.pan
```

Voir aussi

Coercition, pim3d22d

Prototype C++

```
Errc PIm2d23d( const Img2duc &im_in, Img3duc &im_out );
```

Auteur: Jean-Marie Janik

pim2rg

Création d'une carte de régions à partir d'une image d'étiquettes.

Synopsis

pim2rg [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pim2rg** permet de transformer une image d'étiquettes *im_in* en une carte de régions *rg_out*. Chaque valeur de pixel est considérée comme un numéro d'étiquette.
Dans le cas d'images signées, les valeurs négatives ne sont pas prises en compte.

Attention : Il n'y a pas réetiquetage des régions, mais seulement une recopie des numéros de l'image d'étiquettes. Pour un vrai étiquetage voir les opérateurs de segmentation type **plabeling**.

Entrées

- *im_in*: une image.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de régions contenues dans la carte, ou FAILURE en cas d'erreur.

Exemples

Construit la carte de régions à partir du seuillage de l'image tangram.pan:

```
pbinarization 100 1e30 examples/tangram.pan a.pan
pim2rg a.pan b.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PIm2Rg( const Img2duc &im_in, Reg2d &rg_in );
```

Auteur: Régis Clouard

pim2sf

Conversion automatique d'une image de n'importe quel type en type float.

Synopsis

pim2sf [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pim2sf permet de créer une nouvelle image de type float à partir d'une image quelconque. Il n'y a aucune normalisation, il s'agit d'une simple coercition (casting) en utilisant la norme C. Il faut donc préparer les données à cette coercition (avec une normalisation par exemple).

L'opération consiste en:

```
pixel(im_out)=(float)pixel(im_in);
```

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de type Float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image tangram.pan en image de Float:

```
pim2sf a.pan out.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PIm2Sf( const Img2d &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

pim2sl

Conversion automatique d'une image de n'importe quel type en type signed long.

Synopsis

pim2sl [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pim2sl permet de créer une nouvelle image de type signed long à partir d'une image quelconque. Il n'y a aucune normalisation, il s'agit d'une simple coercition (casting) en utilisant la norme C. Il faut donc préparer les données à cette coercition (par une normalisation par exemple).

L'opération consiste en:

```
pixel(im_out)=(long)pixel(im_in);
```

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de type Slong.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image tangram.pan en image signed long:

```
pim2sl tangram.pan out.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PIm2Sl( const Img2d &im_in, Img2dul &im_out );
```

Auteur: Régis Clouard

pim2uc

Conversion automatique d'une image de n'importe quel type en type unsigned char.

Synopsis

pim2uc [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pim2uc permet de créer une nouvelle image de type unsigned char à partir d'une image quelconque. Il n'y a aucune normalisation, il s'agit d'une simple coercition (casting) en utilisant la norme C. Il faut donc préparer les données à cette coercition (par une normalisation par exemple).

L'opération consiste en:

```
pixel(im_out)=(Uchar)pixel(im_in);
```

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de type Uchar.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image de Float a.pan en image de Uchar b.pan:

```
pim2uc a.pan b.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PIm2Uc( comst Img2d &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

pim3d22d

Conversion d'une image ou d'une carte de régions 3D avec un seul plan en image ou carte de régions 2D.

Synopsis

pim3d22d [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pim3d22d** construit une image 2D *im_out* à partir d'une image 3D *im_in* ne possédant qu'un seul plan. Si l'entrée est une carte de régions 3D alors la sortie sera une carte de sortie 2D.

L'opérateur pgetslice peut être utilisé pour extraire un plan d'une image 3D à plusieurs plans.

Entrées

- *im_in*: une image 3D ou une carte de régions 3D.

Sorties

- *im_out*: une image 2D du même type que l'image d'entrée ou une carte de régions 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'images 3D a.pan avec 1 plan en image 2D b.pan:

```
pim3d22d a.pan b.pan
```

Voir aussi

Coercition, pim2d23d

Prototype C++

```
Errc PIm3d22d( const Imc2duc &im_in, Img2duc &im_out, Long noplan );
```

Auteur: Régis Clouard

pimc2img

Construction d'une image de niveaux de gris avec un plan d'une image couleur.

Synopsis

pimc2img *noplan* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pimc2img** permet de créer une nouvelle image de niveaux de gris *im_out* à partir d'un plan d'une image couleur d'entrée *im_in*. Le type des pixels de l'image de sortie *im_out* est le même que celui de l'image d'entrée *im_in*.

Paramètres

- *noplan* est un entier de l'intervalle [0..2] avec la convention suivante:
  - 0: le 1er plan; par exemple le rouge dans une image RGB, ou la teinte dans une image HSL,
  - 1: le 2eme plan; par exemple le vert dans une image RGB, ou la saturation dans une image HSL
  - 2: le 3eme plan; par exemple le bleu dans une image RGB, ou la luminance dans une image HSL.

Entrées

- *im_in*: une image couleur.

Sorties

- *im_out*: une image en niveaux de gris.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Extrait la première bande de l'image couleur parrot.pan:

```
pimc2img 0 parrot.pan a.pn
```

Voir aussi

Coercition

Prototype C++

```
Errc PImc2Img( const Imc2duc &ims, const Img2duc &im2, int plan);
```

Auteur: Régis Clouard

pimc2imx

Construction d'une image multipectrale à 3 bandes à partir d'une image couleur.

Synopsis

pimc2imx [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pimc2imx** permet de créer une image multispectrale à 3 bandes à partir d'une image couleur.

Le type de l'image multispectrale *im_in* est le même que celui de l'image d'entrée *im_in*.

Entrées

- *im_in*: une image couleur.

Sorties

- *im_out*: une image multispectrale.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image couleur parrot.pan en image multispectrale a.pan:

```
imc2imx parrot.pan a.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PImc2Imx( const Imc2duc &im_in, Imx2duc &im_out );
```

Auteur: Régis Clouard

pimg2imc

Conversion d'une image en niveaux de gris en image couleur.

Synopsis

pimg2imc [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pimg2imc permet de convertir une image en niveaux de gris en iage couleur en recopiant l'image en niveaux d egris sur les trois bandes de l'image couleur.

Entrées

- *im_in*: une image en niveaux de gris.

Sorties

- *im_out*: une image couleur.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Transforme l'image en niveaux de gris tangram.pan en image en couleur:

```
pimg2imc examples/tangram.pan out.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PImg2Imc( const Img2duc &im_in, Imc2duc &im_out);
```

Auteur: Régis Clouard

pimg2imx

Conversion d'une image en niveaux de gris en image multispectrale.

Synopsis

pimg2imx *dimension* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pimg2imx** crée une image multispectrale à partir d'une images en niveaux de gris. Le paramètre *dimension* spécifie le nombre de bandes de l'image de sortie. L'image de niveaux de gris est recopiée sur chacune des bandes de l'image de sortie *im_out*.

Param&eagrave;tre

- *dimension* spécifie le nombre de bande de l'image de sortie. C'est un entier positif >0.

Entrées

- *im_in**: une image en niveaux de gris.

Sorties

- *im_out*: une image multispectrale.

Résultat

Retourne SUCCESS ou FAILURE si les images sont incompatibles.

Exemples

Crée une image multispectrale à 2 bandes:

```
pimg2imx 2 tangram.pan a.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PImg2Imx( const Img2duc &im_in, Imx2duc &im_out );
```

Auteur: Régis Clouard

pimgs2imc

Construction d'une image couleur à partir de trois images de niveaux de gris.

Synopsis

pimgs2imc [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_in3*|-] [*im_out*|-]

Description

pimgs2imc permet de créer une image couleur à partir de trois images de niveaux de gris (qui correspondent aux trois composantes couleur).

Les images sont prises dans l'ordre des bandes Rouge, Vert, Bleu.

Entrées

- *im_in*: une image en niveaux de gris.

Sorties

- *im_out*: une image couleur.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit l'image couleur out.pan avec a.pan comme composante rouge, b.pan en composante verte, et c.pan en composante blue.

```
pimgs2imc a.pan b.pan c.pan out.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PImgs2Imc( const Img2duc &im_in1, const Img2duc &im_in2, const
Img2duc &im_in3, Imc2duc &im_out);
```

Auteur: Régis Clouard

pimgs2imx

Création d'une image multspectrale à partir de plusieurs images niveaux de gris.

Synopsis

pimgs2imx *dimension* [*-m mask*] [*im_in*|-]* [*im_out*|-]

Description

L'opérateur **pimgs2imx** crée une image multispectrale à partir d'une liste d'images en niveaux de gris. Le paramètre *dimension* spécifie le nombre d'images mais aussi le nombre de bandes de l'image de sortie *im_out*.

Chaque image correspond à une bande dans l'image multispectrale en gardant l'ordre donné par la liste des arguments de l'opérateur.

Toutes les images d'entrée doivent avoir les mêmes propriétés.

Paramètre

- *dimension* spécifie le nombre d'images d'entrée, C'est un entier positif >0.

Entrées

- *im_in**: une image niveaux de gris.

Sorties

- *im_out*: une image multispectrale.

Résultat

Retourne SUCCESS ou FAILURE si les images sont incompatibles.

Exemples

Crée une image multispectrale à 2 bandes:

```
pimgs2imx 2 tangram.pan lena.pan a.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PImgs2Imx( const Pobjects *im_in[], Imx2duc &im_out );
```

Auteur: Régis Clouard

pimx2imc

Construction d'une image couleur à partir d'une image multispectrale à 3 bandes.

Synopsis

pimx2imc *colorspace* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pimx2imc** convertit l'image multispectrale *im_in* en image couleur *im_out*. Seules les images multispectrales avec au plus 3 bandes peuvent être converties. Si l'image d'entrée a moins de 3 bandes, les autres plans couleurs sont laissés à 0.

L'espace couleur de l'image résultante est précisé par le paramètre *colorspace*.

Paramètres

- *colorspace* est un entier qui spécifie l'espace couleur:
  - 0: RGB
  - 1: XYZ
  - 2: LUV
  - 3: LAB
  - 4: HSL
  - 5: AST
  - 6: I1I2I3
  - 7: LCH
  - 8: WRY
  - 9: RNGNBN
  - 10: YCBCR
  - 11: YCH1CH2
  - 12: YIQ
  - 13: YUV

Entrées

- *im_in*: une image multispectrale.

Sorties

- *im_out*: une image couleur.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image multispectrale a.pan en image couleur RGB b.pan:

```
pimx2imc 0 a.pan b.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PImx2imc( const Imx2duc &im_in, Imc2duc &im_out, int colorspace
);
```

Auteur: Régis Clouard

pimx2img

Construction d'une image de niveaux de gris avec une bande d'une image multispectrale.

Synopsis

pimx2img *nobande* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pimx2img permet de créer une nouvelle image de niveaux de gris *im_out* à partir d'une bande de l'image multispectrale d'entrée *im_in*. Le type des pixels de l'image de sortie *im_out* est le même que celui de l'image d'entrée *im_in*.

Paramètres

- *nobande* est un entier inférieur au nombre de bandes de l'image d'entrée.

Entrées

- *im_in*: une image multispectrale.

Sorties

- *im_out*: une image en niveaux de gris.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Extrait la bande #0 de l'image a.pan:

```
pimx2img 0 a.pan b.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PImx2Img( const Imx2duc &im_in, Img2duc &im_out, int nobande );
```

Auteur: Régis Clouard

pinnermerging

Fusion de régions englobées dans d'autres régions.

Synopsis

pinnermerging [*-m mask*] [*rg_in1*|-] [*rg_in2*|-][*rg_out*|-]

Description

L'opérateur **pinnermerging** fusionne les régions internes. La carte de région de sortie *rg_out* est construite avec toutes les régions de la carte d'entrée *rg_in1* auxquelles ont été fusionnées les régions de la carte *rg_in2* qui sont englobées par celles de *rg_in1*.

Une région de *rg_in2* est englobée dans une région de *rg_in1* si la région de *rg_in1* est le seule région voisine de la région *rg_in2* (le fond est ici considéré comme une région voisine potentielle).

Les régions de *rg_in2* n'ayant pas de voisines dans *rg_in1* ne se retrouvent pas dans le résultat.

Le nombre de région de la carte *rg_out* est le même que celui de *rg_in1* et les numéros de labels sont aussi conservés.

Entrées

- *rg1_in*: une carte de régions.
- *rg2_in*: une carte de régions.
- *gr_in*: un graphe.

Sorties

- *rg_out*: une carte de régions.
- *gr_out*: un graphe.

Résultat

Retourne le nombre de régions fusionnées.

Exemples

Segmente l'image tangram.pan par fusion des régions internes :

```
pbinarization 96 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pfillhole b.pan c.pan
pdif b.pan c.pan d.pan
plabeling 8 d.pan e.pan
pinnermerging b.pan e.pan out.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PInnerMerging( const Reg2d &rg_in1, const Reg2d &rg_in2, Reg2d
&rg_out );
```

Auteur: Régis Clouard

pinnermostmerging

Fusion de régions avec la région voisine la plus englobante.

Synopsis

pinnermostmerging [*-m mask*] [*rg_in1*|-] [*rg_in2*|-] [*rg_out*|-]

Description

L'opérateur **pinnermostmerging** fusionne les régions internes avec les régions les plus englobantes. La carte de régions *rg_out* est construite avec toutes les régions de la carte d'entrée *rg_in1*, auxquelles ont été fusionnées les régions spécifiées de la carte *rg_in2* avec lesquelles elles ont le plus de points de frontière.

Les régions de *rg_in2* n'ayant pas de voisines dans *rg_in1* ne se retrouvent pas dans le résultat.

Le nombre de régions de la carte *rg_out* est le même que celui de *rg_in1* et les numéros de labels sont aussi conservés. Il apparait donc des trous dans la rénumérotation.

Entrées

- *rg_in*: une carte de régions.
- *gr_in*: une carte avec les régions à fusionner.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de régions de *rg_in2* fusionnées.

Exemples

Segmente l'image tangram.pan par fusion des régions internes :

```
pbinarization 96 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pfillhole b.pan c.pan
pdif b.pan c.pan d.pan
plabeling 8 d.pan e.pan
pinnermostmerging b.pan e.pan out.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PInnerMostMerging( const Reg2d &rg_in1, const Reg2d &rg_in2,
Reg2d &rg_out );
```

Auteur: Régis Clouard

pinnerselection

Sélection des régions englobées dans une autre région.

Synopsis

pinnerselection [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur construit la carte de région de sortie *rg_out* avec toutes les régions de la carte d'entrée *rg_in* qui sont entièrement englobées par une seule autre région.

Une région est englobée dans une autre lorsque la région englobée ne possède qu'une seule région voisine: le région englobante.

Une région qui touche le bord ou qui touche le fond (fond=région de label 0) n'est pas considérée comme une région englobée.

Entrées

- *rg_in*: une carte de régions 2D

Sorties

- *rg_out*: une carte de régions 2D

Résultat

Retourne le nombre de régions sélectionnée.

Exemples

Sélectionne les régions internes de la carte rin.pan :

```
pinnerselection rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PInnerSelection( const Reg2d &rg_in, Reg2d &rg_out );
```

Auteur: Régis Clouard

pinsertregion

Insertion de régions dans une image.

Synopsis

pinsertregion [*rg_in*|-] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pinsertregion** construit l'image *im_out* avec les pixels de l'image *im_in2* et les pixels de l'image *im_in1* qui sont dans la bo&icric;te englobante définie par la région donnée dans la carte de régions *rg_in*. Ceci permet de réinsérer une sous-image dans une image à l'endroit spécifié par la boîte englobante des régions.

Entrées

- *rg_in* : une carte de régions.
- *im_in1* : une image ou une carte de régions (inférieure à im_in2).
- *im_in2* : une image ou une carte de régions.

Sorties

- *im_out* : une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Extrait la sous-image autour des pièces de tangram puis la remet dans l'image initiale.

```
pbinarization 87 255 examples/tangram.pan a.pan
plabeling 8 a.pan b.pan
pextractregion b.pan examples/tangram.pan c.pan
pinsertregion b.pan c.pan examples/tangram.pan out.pan
```

Voir aussi

Utilitaire, pinsertregion

Prototype C++

```
Errc PInsertRegion( const Reg2d &rg_in, const Img2duc &im_in1, const
Img2duc &im_in2, Img2duc &im_out );
```

Auteur : Régis Clouard

pinsertsubimage

Insertion d'une image dans une autre image.

Synopsis

pinsertsubimage *x y z* [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pinsertsubimage** permet d'insérer l'image *im_in2* dans l'image *im_in1* aux coordonnées (*x*,*y*, *z*) pour construire l'image de sortie *im_out*. L'insertion se fait en remplaçant les pixels de l'image *im_in1* par les pixels de l'image *im_in2*.

Pour les cartes de régions, les régions de l'image *im_in2* sont ajoutées avec des labels supérieurs à ceux de *im_in1*. Il peut donc y avoir des "trous" dans la numérotation.

Paramètres

- *x, y, z* spécifient les coordonnées du coin supérieur gauche de l'image *im_in2* dans l'image *im_in1*.

  Dans le cas d'images 2D, le paramètre *z* n'est pas utilisé mais doit être donné.

Entrées

- *im_in*: une image.
- *im_in*: une carte de régions.

Sorties

- *im_out*: une image de même type que l'image d'entrée.
- *im_out*: une carte de régions.

Résultat

Pour les images retourne SUCCESS ou FAILURE.
Pour les cartes de régions retourne le nombre de labels dans la carte de sortie.

Exemples

Insère l'image a.pan dans l'image tangram.pan:

```
pinsertsubimage 10 10 0 a.pan tangram.pan b.pan
```

Voir aussi

Utilitaire, pextractsubimage

Prototype C++

```
Errc PInsertSubImage( const Img2duc &im_in1, const Img2duc &im_in2,
Img2duc &im_out, Long cy, Long cx );
```

Auteur: Régis Clouard

pintegralimage

Calcul de l’image intégrale d’une image.

Synopsis

`pintegralimage` -*m mask*] [*im_in*|-] [*col_out*|-]

Description

L’opérateur **pintegralimage** calcule l’image intégrale de l’image d’entrée *im_in*. L’image intégrale permet de facilement calculer la somme à l’intérieur d’une fenêtre de l’image.

L’image intégrale est définie comme suit:

```
output(x,y) = SUM(input(i,j)) où i dans [0..x] et j dans [0..y].
```

Étant donné l’image intégrale obtenue, le calcul de la somme des pixels d’une région de l’image peut être obtenue grâce aux valeurs de l’intégrale aux coordonnées des quatre coins du rectangle: L4+L1 - (L2+L3) (L1 haut gauche, L2 haut droit, L3 bas gauche et L4 bas droit).

Entrées

- *im_in*: une image 2D ou 3D.

Sorties

- *col_out*: une collection avec un tableau de valeur nommé "internal_array".

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule l’image intégrale de tangram.pan.

```
pintegralimage tangram.pan d.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PIntegralImage( const Img2duc &im_in, Collection &col_out );
```

Auteur:Pierre Buyssens

pinterregioncontrast

Calcul du critère de qualité basé sur le contraste inter-régions.

Synopsis

`pinterregioncontrast` [*-m mask*] [*rg_in*|-] [*im_in*|-]

Description

L'opérateur **pinterregioncontrast** calcule le critère de contraste inter-region tel que défini par M. Levine & A. Nazif[*]. Ce critère est basé sur l'idée qu'une bonne segmentation est caractérisée par un fort contraste entre deux régions adjacentes.

Ce critère est proche de 1 quand les régions sont homogènes et proche de 0 quand elles sont hétérogènes.

Le contraste inter-région est calculé comme suit :

$$\text{criterion} = \sum_R [\ AiCi / \text{sum } Ai\ ] \qquad Ci = \text{sum } [\ (lij * |mi - mj|) / (li * (mi + mj))\ ]$$

where

- Ai est la surface de la région .i
- Ci est le contraste de la région i.
- mi est la moyenne de la région i.
- mj est la moyenne de la région j adjacente à i,
- li est le périmetre de la région i.
- lij est la longueur de la frontière entre la région i et la région j.

Attention: Les régions de label=0 ne sont pas prises en compte pour la mesure.

Entrées

- *rg_in*: une carte de régions.
- *im_in*: une image de niveaux de gris.

Résultat

Retourne une valeur réelle positive [0..1].
(Utiliser `pstatus` pour récupérer cette valeur).

Exemples

Calcule la mesure de contaste inter-region pour le cas d’une simple segmentation par binarisation :

pintraregionuniformity

Calcul du critère de qualité basé sur une mesure d'uniformité des régions.

Synopsis

pintraregionuniformity [*-m mask*] [*rg_in*|-] [*im_in*|-]

Description

L'opérateur **pintraregionuniformity** calcule le critère d'uniformité intra-région tel que défini par M. Levine & A. Nazif[*]. Ce critère est basé sur l'idée qu'une bonne segmentation est caractérisée par une forte uniformité intra-région. L'uniformité est calculé à partr du contraste interne des régions.

Ce critère est proche de 1 quand les régions sont homogènes et proche de 0 quand elles sont hétérogènes.

Le critère d'uniformité intra-région st calculé comme suit :

$$1 - \sum_R \left[\sum_s \left(ims[s] - mean(ims) \right)^2 \right]$$

où s est un site (pixel) de l aregion.

Attention: Les régions de label=0 ne sont pas prises en compte pour la mesure.

Entrées

- *rg_in*: une carte de région.
- *im_in*: une image de niveaux de gris.

Résultat

Retourne la valeur du critère [0..1].
(Utiliser `pstatus` pour récupérer cette valeur).

Exemples

Calcule la valeur d'uniformité intra-region pour une simple segmentation par binarisation.

```
pbinarization 80 1e30 tangram.pan i1.pan
plabeling 8 i1.pan i2.pan
pintraregionuniformity i2.pan tangram.pan
pstatus
```

Voir aussi

Evaluation, pinterregioncontrast

Prototype C++

```
Errc PIntraRegionUniformity( const Reg2d &rg_in, const Img2duc
&im_in );
```

Reference

* M. D. Levine and A. M. Nazif, "Dynamic measurement of computer generated image segmentations", *IEEE Trans. PAMI*, 7(2): 155-164, 1985.

Auteur: Régis Clouard

pinverse

Inversion du valeurs de pixel d'une image ou d'un graphe, et inversion des numéros de labels de carte de régions.

Synopsis

pinverse [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pinverse** effectue l'inversion des valeurs de pixel de l'image d'entrée *im_in*.

Pour les images d'octets, l'inversion est appliquée en fonction des valeurs minimale et maximale du type (0-255) :

```
pixel = (max(type) + min(type)) - pixel.
```

Pour les images de réels ou d'entiers longs, l'inversion est faite sur les valeurs minimale et maximale de l'image :

```
pixel = (max(image) + min(image)) - pixel.
```

Pour les images couleur et multispectrales, l'inversion est effectuée sur chaque bande séparément.

Pour les graphes, l'inversion est appliqué sur chaque valeur de noeud :

```
val = (max(graphe) - min(graphe)) - val.
```

Pour les cartes de régions, l'inversion est appliquée sur chaque label non nul :

```
label 0 -> 0;
label 1 -> valeur de label maximum;
label 2 -> valeur de label maximum -1;
...
label maximum -> 1;
```

Entrées

- *im_in*: une image, un graphe ou une carte de régions.

Sorties

- *im_out*: un objet du même type que l'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Inverse les valeurs de pixels de l'image butterfly :

```
inverse examples/butterfly.pan a.pan
```

Voir aussi

Logique

Prototype C++

```
Errc PInverse( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

pjpeg2pan

Conversion d'une image JPEG en image Pandore.

Synopsis

pjpeg2pan *im_in* [*im_out*|-]

Description

L'opérateur **pjpeg2pan** permet de convertir une image de type *JPEG* en un fichier *Pandore*. Le fichier résultant est de type:

- Img2duc pour les images en niveaux de gris,
- Imc2duc pour les images couleur.

Entrées

- *im_in*: un fichier JPEG.

Sorties

- *im_out*: une image Pandore.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pjpeg2pan image.jpeg image.pan
```

Voir aussi

Conversion, ppan2jpeg

Prototype C++

```
Errc PJpeg2Pan( const FILE* fdin, Pobject **objout );
```

Auteur: Régis Clouard

pkmeans

Classification automatique selon les K-moyennes.

Synopsis

pkmeans *attr_in attr_out k maxiter* [*col_in*|-] [*col_out*|-]

Description

L'opérateur `pkmeans` réalise une classification automatique des individus de `col_in` (dont les caractéristiques sont représentées par les tableaux `attr_in.1`, `attr_in.2`, ..., `attr_in.n`) en `K` classes selon l'algorithme des K-moyennes :

Au départ, la moyenne de chaque classe est tirée au hasard (parmi l'ensemble des individus). Puis tous les individus sont affectés à la classe dont la moyenne des caractéristiques est la plus proche. Ce qui permet de calculer de nouvelles moyennes, puis de classer à nouveau les individus. Le processus est répété jusqu'à stabilisation.

Paramètres

- `attr_in` la base du nom des tableaux de caractéristiques des individus à classifier. Ce nom est déclinée. `attr_in.1`, `attr_in.2`, etc. La cellule [j] du tableau `attr_in.i` correspond à la ième caractéristique du j+1ème individu. Ces tableaux sont recherchés dans la collection `col_in`, et doivent tous avoir le même type.
- `attr_out` le nom du tableau d'Ulong (créé dans `col_out`) qui contiendra en sortie le numéro de classe déterminé par l'algorithme pour chaque individu.
- `k` le nombre de classes à rechercher.
- `maxiter` le nombre maximum d'itérations de l'algorithme (pour le cas où il ne stabilise pas).

Entrées

- *col_in*: une collection.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Segmente l’image tangram.pan grâce à la classification par les K-moyennes basée sur la moyenne et la variance des objets:

```
pmeanfiltering 1 tangram.pan moy.pan
pvariancefiltering 0 255 tangram.pan var.pan

pim2array data.1 moy.pan data1.colc
pim2array data.2 var.pan data2.colc
parray2array data.1 Float data1.colc data1.cold
parray2array data.2 Float data2.colc data2.cold
pcolcatenateitem data1.cold data2.cold data3.cold
parraysnorm data data3.cold data3.cold

pkmeans data attrib 5 100 data3.cold cluster.cold

pproperty 0 tangram.pan
w=‘pstatus‘
pproperty 1 tangram.pan
h=‘pstatus‘

parray2im $h $w 0 attrib cluster.Cold kmeans.pan
pim2rg kmeans.pan classif1_out.pan
```

Voir aussi

Classification

Prototype C++

```
Errc PKmeans(const std::string &a_in, const Collection &c_in, const
std::string &a_out, Collection &c_out, int k, int max)
```

Auteur: Alexandre Duret-Lutz

pknn

Classification selon les K plus proches voisins.

Synopsis

pknn *attr_base attr_in attr_out k* [*col_base*|-] [*col_in*|-] [*col_out*|-]

Description

L'opépateur `pknn` effectue une classification selon les K plus proches voisins. Un individu est classé selon la classe majoritaire parmi ses K plus proches voisins de l'espace d'apprentissage.

La mesure de distance entre deux objets xi and xj utilise la distance euclidienne :

$$D_{ij} = [\text{SUM}_{\{d=1;n\}} (xid - xjd)^2]^{1/2}$$

où xid est la caractéristique d de l'objet i et xjd est la caractéristique d for de l'object j.

Paramètres

- *attr_base* est le nom de base des vecteurs de caractéristiques des individus déjà classés. Les n caractéristiques de chaque individu sont stockées sous forme de tableaux `attr_base.1`, `attr_base.2`, ..., `attr_base.n`, tous de même type. Le tableau `attr_base.C` doit contenir les numéros de classe de chaque individu. Si ce tableau est absent, la classe du i-ème individu est i.
- *attr_in* est le nom de base des vecteurs de caractéristiques à classer. Les caractéristiques de chaque individu doivent se trouver dans les tableaux `attr_in.1`, `attr_in.2`, ..., `attr_in.n`.
- *attr_out* est le nom du vecteur d'Ulong qui contient les classes déterminées pour chaque individu de `col_in`.
- `k` est le nombre de voisins à considérer pour la classification.

Entrées

- *col_base*: une collection contenant les paramètres de la classification.
- *col_in*: une collection à classer.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Classification des bonbons de l'image jellybean.pan à partir d'un échantillon des différents types de bonbons stockés dans le dossier 'base' (Unix version):

```
# Learning
   classes=1;
   for i in base/*.pan
   do
      pim2array ind $i /tmp/tmp1
      parraysize ind.1 /tmp/tmp1
      size=‘pstatus‘
      pcreatearray ind.C Ushort $size $classes | pcolcatenateitem /tmp/tmp1 - i-01.pan
      if [ -f base.pan ]
      then pcolcatenateitem i-01.pan base.pan base.pan
      else cp i-01.pan base.pan
      fi
      classes=‘expr $classes + 1‘
   done

# Classification
   pproperty 0 jellybeans.pan
   ncol=‘pstatus‘
   pproperty 1 jellybeans.pan
   nrow=‘pstatus‘

   pim2array ind jellybeans.pan | pknn ind ind ind 10 base.pan - | parray2im $ncol $nrow 0 ind | pim2rg - out.pan
```

Voir aussi

Classification

Prototype C++

```
Errc PKnn(const std::string &a_base, const Collection &c_base, const
std::string &a_in, const Collection &c_in, const std::string &a_out,
Collection &c_out, int K);
```

Auteur: Alexandre Duret-Lutz

plab2lch

Changement d'espace couleur de Lab vers LCH.

Synopsis

plab2lch [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plab2lch** permet de passer de l'espace couleur LAB (Luminancy, red/blue chrominancy, yellow/blue chrominancy) à l'espace LCH (Light, Chroma, Hue). LCH est une version perceptuelle de l'espace HSL.

Entrées

- *im_in*: une image couleur Lab.

Sorties

- *im_out*: une image couleur LCH.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertirt l'image couleur a.pan de l'espace couleur Lab à l'espace LCH

```
plab2lch a.pan b.pan
```

Voir aussi

Color

Prototype C++

```
Errc PLAB2LCH( const Imc2duc &im_in, Imc2duc &im_out );
```

Auteur: Olivier Lezoray

plabeling

Etiquetage des régions homogènes d'une image.

Synopsis

plabeling *connexity* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **plabeling** consiste à marquer avec un même label un ensemble de pixels connexes de *im_in* ayant exactement la même valeur de niveau de gris.

Un pixel P1 et un pixel P2 auront la même valeur de label:

- si P1 est dans le voisinage de P2 (selon la connexité choisie)
- et si valeur(P1) = valeur(P2).

Une région de la carte de régions *rg_out* est définie par une ensemble de pixels ayant exactement le même label. Chaque région reçoit un numéro de label unique et minimal mais non nul. Les valeurs de label sont attribuées au hasard: La première région formée reçoit le numéro 1, la suivante 2 etc. Il n'y a pas de trou dans la numérotation, toutes les régions entre 1 et le nombre de labels sont attribuées.

Si *im_in* est une carte de régions, alors **plabeling** permet de renuméroter la carte de régions, pour minimiser le nombre de labels en comblant les trous dans la numérotation.

Remarque: le label 0 à une sémantique particulière dans Pandore. Une région de label 0 est considérée comme une non-région.

Paramètres

- *connexity* définit la notion de voisinage. Les valeurs possibles sont:
  - 8 ou 4 voisinage en 2D;
  - 6 ou 26 voisinage en 3D.

Entrées

- *im_in*: une image de niveaux de gris, une carte de régions ou un graphe.

Sorties

- *rg_out*: une carte de régions ou un graphe.

Exemples

Squelette par zone d'influence (le skiz).

```
pbinarization 100 1e30 tangram.pan i1.pan
pdistance i1.pan i2.pan
plabeling 8 i1.pan i3.pan
pwatershed i3.pan i2.pan i4.pan
pboundary 8 i4.pan out.pan
```

Résultat

Retourne le nombre de régions construites
ou FAILURE en cas d'erreur.

Voir aussi

Segmentation

Prototype C++

```
Errc PLabeling( const Img2duc &im_in, Reg2d &rg_out, int connexity
);
```

Auteur: Régis Clouard

plabelmasking

Sélection de régions spécifiées une autre carte de régions.

Synopsis

plabelmasking [*rg_in1*|-] [*rg_in2*|-] [*rg_out*|-]

Description

L'opérateur

plabelmasking selects the regions from the region map <>rg_in1 if one label of that region is hit by one label of the regions in *rg_in2*.

Entrées

- *rg_in1*: une carte de régions
- *rg_in2*: une carte de régions utilisée comme masque.

Sorties

- *rg_out*: une carte de régions

Résultat

SUCCESS ou FAILURE.

Exemples

Selects the regions of rin1.pan specified in the region map rin2.pan:

```
plabelmasking rin1.pn rin2 rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PlabelMasking( const Reg3d &rg_in1, const Reg3d &rg_in2, Reg3d &rg_out );
```

Auteur: Régis Clouard

plabelmerging

Fusion nominative de 2 régions.

Synopsis

plabelmerging *label1 label2* [*rg_in*|-] [*gr_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

L'opérateur **plabelmerging** effectue la fusion de deux régions identifiées par leur numéro de label. En sortie, la région résultante conserve le numéro *label1*. Dans le graphe associé, les coordonnées du nouveau sommet sont placées au milieu des 2 sommets fusionnés. De plus, la liste des voisins de chacun des sommets est fusionnée dans le nouveau.

Paramètres

- *label1* et *label2* sont des numéros de label valides dans la carte *rg_in*.

Entrées

- *rg_in*: une carte de régions.
- *gr_in*: un graphe.

Sorties

- *rg_out*: une carte de régions.
- *gr_out*: un graphe.

Résultat

Retourne SUCCESS ou FAILURE selon que la fusion est réussie ou pas.

Exemples

Segmente l'image tangram.pan puis fusionne les deux labels 1 et 2 :

```
puniformityquadtree 0.9 tangram.pan a.pan
prg2gr a.pan b.pan
plabelmerging  1 2 a.pan b.pan out1.pan out2.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PLabelMerging( const Reg2d &rg_in, cconst Graph2d &gr_in, Reg2d
&rg_out Graph2d &gr_out, int label1, int label2 );
```

Auteur: Régis Clouard

plabelselection

Sélection d'une région par son numéro de label.

Synopsis

plabelselection *label* [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **plabelselection** construit la carte de région *rg_out* avec la seule région de numéro *label* extraite de *im_in*, ou la dernière région si *label*=-1 ou supéreiur au nombre de régions total.

Paramètres

- Le *label* est le numéro de région supprimée.
  Si *label* =-1 ou supérieur au nombre de labels alors c'est la dernière région qui est supprimée.

Entrées

- *rg_in*: une carte de régions

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de pixels de la région sélectionnée.

Exemples

Sélectionne la région no. 10 de la carte rin.pan:

```
plabelselection 10 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PLabelSelection( const Reg3d &rg_in, Reg3d &rg_out, Long label
);
```

Auteur: Régis Clouard

plabelsselection

Sélection des régions présentes dans une autre carte de régions.

Synopsis

plabelsselection [*rg_in1*|-] [*rg_in2*|-] [*rg_out*|-]

Description

L'opérateur **plabelsselection** construit la carte de région *rg_out* avec les régions de la carte de régions *rg_in1* qui sont présentes dans la carte de régions *rg_in2*.

Seules les valeurs de label sont utilisées pas la position des régions.

Entrées

- *rg_in1*: une carte de régions
- *rg_in2*: une carte de régions

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de pixels de la région sélectionnée.

Exemples

Sélectionne les régions de la carte rin1.pan qui sont aussi présentes dans la carte rin2:

```
plabelsselection rin.pan rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PLabelsSelection( const Reg2d &rg_in1, const Reg2d &rg_in2,
Reg3d &rg_out );
```

Auteur: Régis Clouard

planczosrescale

Retaille d'une image par l'algorithme de Lanczos.

Synopsis

planczosrescale *rescalex rescaley rescaleyz* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **planczosrescale** utilise un noyau de convolution pour interpoler les valeurs des pixels de l'image d'entrée *im_in* afin de calculer les valeurs des pixels de l'image de sortie *im_out*.
L'interpolation consiste à pondérer l'influence des pixels d'entrée. Les poids sont dépendants de la position du pixel de sortie et sont donnés par l'algorithme de Lanczos:

```
       | 1 si x=0
L(x) = | sinc(x).sin(x/a) si -a < x < a
       | 0 sinon
```

Par exemple, si l'image est zoomée par 3, alors chaque pixel de sortie est donné par:

```
for i in [-3, 3]
  for j in [-3, 3]
      im_out[p.y][p.x] += L(i*scalex)*L(j*scaley)*im_in[p.y*scaley+j][p.x*scalex+i]
```

Pour zoomer une carte de régions ou un graphe, il faut utiliser l'operateur prescale.

Paramètres

- *rescalex*, *rescaley*, *rescalez* sont des réels positifs correspondant aux facteurs de retaille.
  Si les rescales sont > 1 alors il s'agit d'un agrandissement, s'ils sont < 1 alors il s'agit d'une réduction.
  rescalez est ignoré pour le cas des images 2D mais doit être donné.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Agrandissement de l'image d'un facteur 2 :

```
planczosrescale 2 2 0 tangram.pan a.pan
```

Réduction de l'image d'un facteur 2 :

```
planczosrescale 0.5 0.5 0 tangram.pan a.pan
```

Voir aussi

Transformation, plinearrescale, pbicubicrescale, pbellrescale, pmitchellrescale, pquadraticbsplinerescale, prescale

Prototype C++

```
Errc PLanczosRescale( const Img2duc &im_in, Img2duc &im_out, float rescaley, float rescalex );
```

Auteur: Régis Clouard

plaplacian

Approximation du Laplacian d'une image.

Synopsis

plaplacian *connexity* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plaplacian** permet d'approximer le calcul de l'amplitude de la derivée seconde de l'image *im_out*.

Cet opérateur est notamment utilisé pour détecter les contours des objets. L'intérêt, c'est qu'il donne des contours fermés. Par contre, il est très sensible au bruit.

L'algorithme consiste à convoluer l'image par le masque:
pour une *connexity*=4:

```
|+0 -1  +0|
|-1  +4 -1|
|+0 -1  +0|
```

pour une *connexity*=8:

```
|-1 -1- 1|
|-1  +8 -1|
|-1 -1 -1|
```

L'image de sortie *im_out* est de même type que l'image d'entrée *im_in*.

Pour les images de Uchar (unsigned char), les valeurs sont décalées de 127 (le 0 devient 127). Ainsi, pour la détection des passages par 0, il faut donc utiliser l'opérateur pzerocross avec la valeur 127.

Pour les images de Slong (signed long), les valeurs ne sont pas décalées. Pour la détection des passages par 0, il faut donc utiliser l'opérateur pzerocross avec la valeur 0.

Paramètres

- *connexity* spécifie le type de maillage: soit un voisinage 4 soit 8.

Entrées

- *im_in*: une image 2D de niveaux de gris.

Sorties

- *im_out*: une image du même type que l'image d'entrée im_in.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Détection de contours selon l'algorithme LOG (Laplacien de Gaussiennes)

```
pgaussfiltering 1.2 tangram.pan a.pan
plaplacian 8  a.pan b.pan
pzerocross 8 127 b.pan out.pan
```

Voir aussi

Détection de contours, pzerocross

Prototype C++

```
Errc PLaplacian( const Img2duc &im_in, Img2duc &im_out, int
connexity );
```

Auteur: Régis Clouard

pleafcutting

Suppression des feuilles d'un graphe.

Synopsis

pleafcutting [*-m mask*] [*gr_in*|-] [*gr_out*|-]

Description

L'opérateur **pleafcutting** procède à la suppression physique des arcs dont l'un des sommets de ses extremités ne possède qu'un voisin (ie, est une feuille). Le sommet isolé est tout de même conservé dans le graphe de sortie *gr_out*.

Entrées

- *gr_in*: un graphe.

Sorties

- *gr_out*: un graphe.

Résultat

Retourne le nombre d'arcs supprimés.

Exemples

```
pleafcutting g1.pan g2.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PLeafCutting( const Graph &gr_in, Graph &gr_out );
```

Auteur: François Angot

plegendrepolynomialfitting

Calcul de l'approximation du fond d'une image en utilisant une approximation par polynômes de Legendre.

Synopsis

plegendrepolynomialfitting [*im_in*|-] [*im_out*|-]

Description

plegendrepolynomialfitting convertit le contenu d'une image en un fond homogène en utilisant une approximation par polynômes de Legendre.

Il utilise la relation d'orthogonalité des polynômes de Legendre pour construire l'image comme la double somme de ces fonctions. La somme est ensuite évaluée pour produire l'image qui approxime une projection.

Paramètres

- *xorder* donne le degré pour x [0..10].
- *yorder* done le degré pour y [0..10].

Plus les valeurs de degré sont élevées plus l'approximation est faible (plus proche de l'image originale).

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: une image 2D de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Correction de l'illumination de l'image tangram en utilisant la soustraction de fond. Le fond d'images est approximé par une expression polynomiale d'ordre 2:

```
plegendrepolynomialfitting 2 2 tangram.pan a.pan
psub tangram.pan a.pan b.pan
pmeanvalue a.pan; mean=‘pstatus‘
paddcst $mean b.pan out.pan
```

Autres exemples

Voir aussi

Surface Fitting

Prototype C++

```
Errc PLegendrePolynomialFitting( const Imx2d &im_in, Imx2d &im_out,
int xOrder, int yOrder );
```

Auteur: Régis Clouard

plineardilation

Dilatation morphologique des points de plus fort contraste d'une image par une ligne.

Synopsis

plineardilation *orientation1 orientation2 halfsize* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plineardilation** permet de dilater les points de plus fort contraste avec un élément structurant linéaire de direction *orientation1* dans le plan x,y et *orientation2* dans l'espace x,y,z.

La taille de l'élément structurant est de *halfsize* pixels de part et d'autre du pixel central.

La dilatation correspond à l'opération: remplacer le pixel central par la valeur maximale de ses voisins.

```
dilatation(x,y)= Max(voisins selon l'élément structurant de x,y).
```

Pour une image binaire cela revient à dilater les régions blanches.

Pour les cartes de régions, la dilatation ne dilate que les bords de régions qui touchent le fond.

Paramètres

- *orientation1* correspond à l'angle exprimé en degrés de la ligne dans le plan x,y.
- *orientation2* correspond à l'angle exprimé en degrés de la ligne dans l'espace entre z et le plan x,y.
- *halfsize* est la demi-taille de l'élément structurant en nombre de pixels. La ligne a donc une taille de 2*demitaille+1.

Entrées

- *im_in*: une image ou une carte de régions.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Récupération des parties de frontière de pièces de tangram qui ont au moins 5 pixels verticaux.

```
pgradient 1 tangram.pan i1.pan i2.pan
pbinarization 20 -1 i1.pan i3.pan
plinearerosion 2 5 i3.pan i4.pan
plineardilation 2 5 i4.pan out.pan
```

Voir aussi

Morphologie, plinearerosion.

Prototype C++

```
Errc PLinearDilation( const Img2duc &im_in, Img2duc &im_out, int
orientation, int halsize );
```

Auteur: Régis Clouard

plinearerosion

Erosion morphologique des points de plus fort contraste d'une image par une ligne.

Synopsis

plinearerosion *orientation1 orientation2 halfsize* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plinearerosion** permet d'éroder les points de plus fort contraste avec un élément structurant linéaire de direction *orientation1* dans le plan x,y et *orientation2* dans l'espace x,y,z.

La taille de l'élément structurant est de *halfsize* pixels de part et d'autre du pixel central.

L'érosion correspond à l'opération : remplacer le pixel central par la valeur minimale de ses voisins.

```
erosion(x,y)= Min(voisins selon l'élément structurant de x,y).
```

Pour une image binaire cela revient à éroder les régions blanches.

Pour les cartes de régions, l'érosion ajoute des labels nuls aux points d'érosion.

Paramètres

- *orientation1* correspond à l'angle exprimé en degrés de la ligne dans le plan x,y.
- *orientation2* correspond à l'angle exprimé en degrés de la ligne dans l l'espace entre z et la plan x,y.
- *halfsize* est la demi-taille de l'élément structurant en nombre de pixels. La ligne a donc une taille de 2*demitaille+1.

Entrées

- *im_in*: une image ou une carte de régions.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Récupération des parties de frontière de pièces de tangram qui ont au moins 5 pixels verticaux.

```
pgradient 1 tangram.pan i1.pan i2.pan
pbinarization 20 -1 i1.pan i3.pan
plinearerosion 90 0 5 i3.pan i4.pan
plineardilation 90 0 5 i4.pan out.pan
```

Voir aussi

Morphologie, plineardilation.

Prototype C++

```
Errc PLinearErosion( const Img2duc &im_in, Img2duc &im_out, int
orientation1, int orientation2, int halfsize );
```

Auteur: Régis Clouard

plinearinterpolation

Remplacement de pixels manquants par interpolation linéaire des voisins.

Synopsis

plinearinterpolation *window_depth window_height window_width* [*im_in*|-] [*im_msk*|-] [*im_out*|-]

Description

L'opérateur **plinearinterpolation** permet de remplacer les pixels de l'image *im_in* qui sont masqués dans l'image *im_msk* par interpolation linéaire des pixels du voisinage. La voisinage d'un pixel est défini par une fenêtre de taille *window_depth x window_height x window_width* centrée sur le pixel.

L'image de sortie *im_out* est construite ainsi :

```
si im_msk(y,x) = 0 alors im_out(y,x) = im_in(y,x)
                   sinon im_out(y,x) = mean(im_in, (x, y)), window)
```

où mean(im_in, (y,x), window) est la moyenne des pixels non masqués dans la fenêtre autour du point (x,y).

Paramètres

- *window_width*, *window_height*, *window_depth* définissent la taille du voisinage de l'interpolation.

Entrées

- *im_in* : une image.
- *im_msk* : une image binaire.

Sorties

- *im_out* : une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit un masque rond et applique l'interpolation linéaire à l'intérieur du rond :

```
pshapedesign 205 244 0 1 20 0 a.pan
plinearinterpolation 20 20 0 examples/butterfly.pan a.pan b.pan
```

Voir aussi

Interpolation

Prototype C++

```
Errc PLinearInterpolation( const Img2duc &im_in;, const Img2duc
&im_msk, Img2duc &im_out, int window_depth, int window_height, int
window_width );
```

Auteur: Régis Clouard

plinearregression

Calcul de l'approximation du fond d'une image en utilisant la regression linéaire.

Synopsis

plinearregression [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plinearregression** convertit le contenu d'une image en un fond d'image uniforme, en utilisant la regresion linéaire. Cet opérateur supprime toute sorte de défaut d'illumination avec un changement graduel de couleur de fond entre un bord d'image et son opposé. Cet opérateur n'est pas adapté à la suppression de tâches d'illumination circulaires (type spotlight).

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: une image 2D de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Correction de l'illumination de l'image de tangram en utilisant la soustraction du fond estimé à partir de la régression liénaire:

```
plinearregression tangram.pan a.pan
psub tangram.pan a.pan b.pan
pmeanvalue a.pan; mean='pstatus'
paddcst $mean b.pan out.pan
```

Autres exemples

Voir aussi

Surface Fitting

Prototype C++

```
Errc PLinearRegression( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

plinearrescale

Augmentation ou réduction de la taille d'une image par interpolation bilinéaire.

Synopsis

plinearrescale *rescalex rescaley rescaleyz* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plinearrescale** permet l'agrandissement ou la réduction de la taille d'une image par un facteur *rescalex* selon l'axe x, *rescaley* selon l'axe y et *rescalez* selon l'axe z (pour les images 3D). L'image est agrandie selon un axe si le facteur de rescale est > 1 et réduite si le facteur de rescale est >0 et <1.

Cette version utilise l'interpolation bilinéaire. L'interpolation bilinéaire utilise une moyenne pondérée dans le voisinage 2x2 du pixel de l'image d'entrée pour déterminer la valeur du pixel de sortie :

```
sx = (x/rescalex)-||x/rescalex||
sy = (x/rescaley)-||y/rescaley||
dx = sx - ||sx||
dy = sy - ||sy||
im_out[y][x] = ((1-dx) * (1-dy) * ims[b][sy][sx]
             + (1-dx)*dy * ims[b][sy+1][sx]
             + dx * (1y-dy) * ims[b][sy][sx+1]
             + dx * dy * ims[b][sy+1][sx+1]);
```

L'interpolation bilinéaire offre un bon compromis entre qualité des résultats et temps d'exécution. Pour les images 2D, de meilleurs résultats peuvent être obtenus avec l'interpolation bicubique mais au prix d'un temps de calcul beaucoup plus long (voir pbicubicrescale).

Pour retailler une carte de régions ou un graphe, il faut utiliser l'opérateur prescale.

Paramètres

- *rescalex*, *rescaley*, *rescalez* sont des réels positifs correspondant aux facteurs de retaille.
  Si les rescales sont > 1 alors il s'agit d'un agrandissement, s'ils sont < 1 alors le rescale il s'agit d'une réduction.
  rescalez est ignoré pour le cas des images 2D mais doit être donné.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Agrandissement de l'image d'un facteur 2 :

```
plinearrescale 2 2 0 tangram.pan a.pan
```

Réduction de l'image d'un facteur 2 :

```
plinearrescale 0.5 0.5 0 tangram.pan a.pan
```

Voir aussi

Transformation, pbicubicrescale, prescale

Prototype C++

```
Errc PLinearRescale( const Img2duc &im_in, Img2duc &im_out, float
rescaley, float rescalex );
```

Auteur: Régis Clouard

plineartransform

Transformation linéaire des niveaux de gris.

Synopsis

plineartransform *inverse min max* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

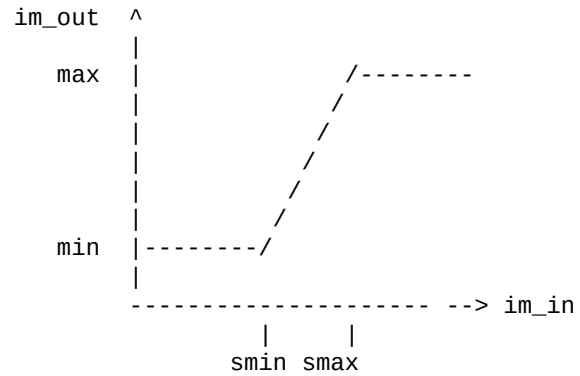
L'opérateur **plineartransform** étale ou compresse la plage des niveaux de gris de l'image d'entrée *im_in* en utilisant une transformation linéaire. Le paramètre *inverse* spécifie si la transformation est positive (*inverse*=0) ou négative (*inverse*=1).

L'effet d'une transformation positive est d'étaler les niveaux de gris entre les nouvelles bornes [min..max].
L'effet d'une transformation négative est d'étaler les niveaux de gris entre les nouvelles bornes [min,max] puis d'inverser les valeurs de niveaux de gris: max devient min, min devient max, etc...

La transformation positive d'un pixel 'p' prend la forme :

```
im_out[p]=(c*(im_in[p]-smin)) + min;
c=(max-min) / (smax-smin)
```

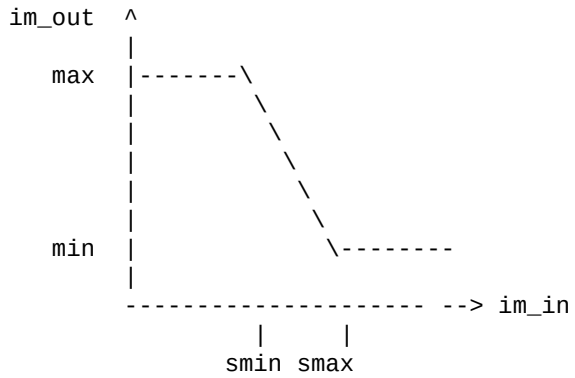
où smin et smax sont respectivement les valeurs minimale et maximale des pixels d'entrée, et c est un facteur de normalisation pour étaler linéairement les niveaux de gris entre *min* et *max*.



La transformation négative d'un pixel 'p' prend la forme :

```
im_out[p]=(c*(smax-ims[p])) + min;
c=(max-min) / (smax-smin)
```

où smin et smax sont respectivement les valeurs minimale et maximale des pixels d'entrée, et c est un facteur de normalisation pour étaler linéairement les niveaux de gris entre *min* et *max*.



Pour les images couleur et multispectrales, la transformation utilise l'approche vectorielle : le min et le max sont calculés sur toutes les bandes et chaque bande est modifiée avec la même transformation.

Paramètres

- *inverse* est un entier dans [0,1] qui spécifie si la transformation est positive (*inverse*=0) ou négative (*inverse*=1).

- *min* et *max* spécifient les bornes des valeurs de pixel en sortie. Les valeurs possibles sont dépendantes du type de l'image d'entrée.
  Note: si *min* < *max* alors min et max sont égaux respectivement aux valeurs minimale et maximale possibles du type (eg. 0 and 255 pour les images Uchar).

Entrées

- *im_in*: an image.

Sorties

- *im_out*: une image avec les mêmes propriétés que l'image d'entrée *im_in*.

Résultat

retourne SUCCESS ou FAILURE en cas de mauvais paramétrage.

Exemples

- Applique une transformation positive puis une transformation négative sur l'image tangram.pan pour créer l'image b.pan. Parce que la transformation négative est l'inverse de la transformation positive, l'image de sortie est la même que l'image d'entrée (aux erreurs d'arrondi près). La différence des deux images doit être nulle:

  ```
  plineartransform 1 0 255 tangram.pan a.pan
  plineartransform 0 0 255 a.pan b.pan
  pdif a.pan b.pan c.pan
  ```

- Applique une transformation linéaire pour crée. l'image a.pan en utilisant les bornes de sortie maximale du type de l'image:

```
plineartransform 0 1 -1 tangram.pan a.pan
```

- Transformation linéaire par morceaux: les pixels d'entrée de l'intervalle [0, 75] sont comprimés dans la nouvelle plage [0..20] et ceux de l'intervalle [76, 255] sont étalés sur la plage [21, 255]:

  pex

```
pthreshold 0 75 tangram.pan a.pan
plineartransform 0 0 20 a.pan a1.pan
paddcst -75 tangram.pan a.pan
plineartransform 0 0 235 a.pan b.pan
paddcst 20 b.pan a2.pan
por a1.pan a2.pan a.pan
```

Voir aussi

Transformation de la LUT, plogtransform, ppowerlawtransform

Prototype C++

```
Errc PLinearTransform( const Img2duc &im_in, const Img2duc &im_out,
int inverse, float min, float max );
```

Auteur: Régis Clouard

plipadd

Addition de 2 images selon le modèle LIP.

Synopsis

plipadd [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **plipadd** calcule la somme des valeurs de niveaux de gris entre chaque pixel de l'image *im_in1* et de l'image *im_in2* correspondant, selon le modèle LIP (Logarithmic Image Processing).

L'addition selon le modèle LIP est définie par :

```
im_out(x,y) = im_in1(x,y) + im_in2(x,y) - [ (im_in1(x,y).im_in2(x,y)) / M]
```

où M est le nombre de niveaux de gris maximal (p. ex. 256 pour une image d'octets). Les deux images d'entrée *im_in1* et *im_in2* doivent être de même type. Au besoin, il est nécessaire d'utiliser les opérateurs de coercition.

L'image de sortie est de même type que les images d'entrée.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes (approche marginale).

Entrées

- *im_in1*: une image.
- *im_in2*: une image.

Sorties

- *im_out*: une image du même type que les images d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Addition de deux images a.pan et b.pan avec résultat dans result.pan:

```
plipadd a.pan b.pan result.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PLipAdd( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc &im_out );
```

Auteur: Régis Clouard

plipmultcst

Multiplication d'une image par une constante selon le modèle LIP.

Synopsis

plipmultcst *cst* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plipmultcst** calcule l'image *im_out* par multiplication des valeurs de pixels de l'image *im_in* par la valeur *cst*, selon le modèle LIP (Logarithmic Image Processing).

La multiplication par une constrante selon le modèle LIP est définie par :

```
im_out(x,y) = M - M.[1-(im_in(x,y)/M)]^cst;
```

où M est le nombre de niveaux de gris maximal des images (eg, 256 pour les image d'octets).

L'image de sortie est de même type que l'image d'entrée.

Paramètres

- *cst* est une valeur réelle.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Division des pixels de l'image tangram.pan par 2 :

```
plipmultcst 0.5 tangram.pan a.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PLipMultCst( const Img2duc &im_in, Img2duc &im_out, float cst
);
```

Auteur: Régis Clouard

plipsub

Soustraction de 2 images selon le modèle LIP.

Synopsis

plipsub [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **plipsub** effectue une soustraction des valeurs de niveaux de gris entre chaque pixel de l'image *im_in1* et de l'image *im_in2* correspondant, selon le modèle LIP (Logarithmic Image Processing).

La soustraction selon le modèle LIP est définie par :

```
im_out(x,y) = M.(im_in1(x,y) - im_in2(x,y)) / ( M - im_in2(x,y))
```

où M est le nombre de niveaux de gris maximal (p. ex. 256 pour une image d'octets).

Les deux images d'entrée *im_in1* et *im_in2* doivent être de même type. Au besoin, il est nécessaire d'utiliser les opérateurs de coercition.

L'image de sortie est de même type que les images d'entrée.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes (approche marginale).

Entrées

- *im_in1*: une image.
- *im_in2*: une image.

Sorties

- *im_out*: une image du même type que les images d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Soustraction de deux images a.pan et b.pan avec résultat dans result.pan:

```
plipsub a.pan b.pan result.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PLipSub( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc &im_out );
```

Auteur: Régis Clouard

plocalextrema

Localisation des points constituant un extréma dans au moins une direction.

Synopsis

plocalextrema *connexity* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plocalextrema** construit l'image des points constituants des extréma dans une des directions possibles selon la *connexité* specifiée.

Un point est extrémum selon une direction s'il est supérieur à au moins un de ses deux voisins opposés dans une direction et pas inférieur à aucun de ses voisins. Par exemple, le point (y,x) est extremum si:

```
im_in[y][x] > im_in[y-1][x] et im_in[y][x] >= im_in[y+1][x]
ou
im_in[y][x] >= im_in[y-1][x] et im_in[y][x] > im_in[y+1][x]
```

Les valeurs des pixels des points extréma sont conservées dans l'image de sortie *im_out*, les autres sont mises à 0. L'image de sortie est donc du même type que l'image d'entrée.

Paramètres

- *connexity* spécifie la relation de voisinage (4, 8 in 2D) or (6, 26 in 3D). Pour les graphes ce paramètre est ignoré.

Entrées

- *im_in*:

Sorties

- *im_out*: une image de niveaux de gris.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Détecte les extréma de l’image tangram.pan:

```
plocalextrema 8 tangram.pan a.pan
```

Voir aussi

Caractérisation image

Prototype C++

```
Errc PlocalExtrema( const Img2duc &im_in, Img2duc &im_out, int
connexity );
```

Auteur: Régis Clouard

plocalmaxima

Localisation des points constituant un maximum local.

Synopsis

plocalmaxima *connexity* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plocalmaxima** construit l'image des points maxima locaux selon la *connexité* donnée.

Un point est maximal s'il ne possède aucun voisin plus grand que lui. Les valeurs des sommets sont conservées, dans l'image de sortie *im_out*.

Pour les graphes, les maxima sont détectés sur les valeurs de sommet.

Paramètres

- *connexity* définit la notion de voisinage (4, 8 en 2D et 6, 26 en 3D). Pour les graphes, ce paramètre est ignoré mais doit être donné.

Entrées

- *im_in*: une image de niveaux de gris ou un graphe.

Sorties

- *im_out*: une image de niveaux de gris ou un graphe.

Résultat

Retourne SUCCCESS ou FAILURE.

Exemples

Localise les maxima locaux de l'image tangram.pan:

```
plocalmaxima 8 tangram.pan a.pan
```

Voir aussi

Caractérisation image, plocalminima

Prototype C++

```
Errc MaximaLocaux( const Img2duc &im_in, Img2duc &im_out, int
connexity );
```

Auteur: Régis Clouard

plocalminima

Localisation des points constituant un minimum local.

Synopsis

plocalminima *connexity* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plocalminima** construit l'image des points minima locaux selon la *connexité* donnée.

Un point est minimal s'il ne possède aucun voisin plus petit que lui. Les valeurs des sommets sont conservées, dans l'image de sortie *im_out*.

Paramètres

- *connexity* définit la notion de voisinage (4, 8 en 2D) ou (6, 26 en 3D). Pour les graphes, ce paramètre est ignoré mais doit être donné.

Entrées

- *im_in*: une image de niveaux de gris ou un graphe.

Sorties

- *im_out*: une image de niveaux de gris ou un graphe.

Exemples

Localise les minima locaux de l'image tangram.pan:

```
plocalminima 8 examples/tangram.pan a.pan
```

Résultat

Retourne SUCCESS ou FAILURE.

Voir aussi

Caractérisation image, plocalmaxima

Prototype C++

```
Errc PLocalMinima( const Img2duc &im_in, Img2duc &im_out, int
connexity );
```

Auteur: Régis Clouard

plocationselection

Sélection de régions sur leur valeur de compacité.

Synopsis

plocationselection *relation xmin ymin zmin xmax ymax zmax* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **plocationselection** permet de sélectionner les régions à partir de leur coordonnées. Les paramètres définissent le parallèlépide de sélection. Si la valeur du paramètre *relation* est positive alors les régions qui sont complètement à l'intérieur du parallèlépide sont sélectionnées. Si la valeur est négative, les régions non complètement à l'intérieur sont sélectionnées.

Attention, il n'y a pas de reétiquetage ds régions.

Paramètres

- *relation* est une valeur positive ou négative. Si la valeur est positive, les régions à l'intérieur du parallèlépide donnée sont gardées. Si la valeur est négative, les régions à l'extérieur sont conservées dans le résultat.

Entrées

- *rg_in*: une carte de régions

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Exemples

Sélectionne les régions de l'image *tangram.pan* à l'intérieur du rectangle (10,0, 238, 228):

```
pbinarization 112 256 examples/tangram.pan b.pan
plabeling 8 b.pan r.pan
plocationselection -1 0 10 0 238 228 0 r.pan v.pan
```

Voir aussi

Région

Prototype C++

```
Errc PLocationSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, int xmin, int ymin, int xmax, int ymax);
```

Auteur: Régis Clouard

plog

Logarithme népérien d'une image or d'un graphe.

Synopsis

plog [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plog** construit le logarithme népérien de l'entrée *im_in*. Si l'entrée est une image, chaque pixel de l'image de sortie *im_out* est construit avec le logarithme du pixel correspondant dans l'image d'entrée *im_in*.

Le cas du 0 est géré en utilisant un epsilon. La formule de calcul est donc :

```
if (pixel(im_in) == 0 )
   pixel(im_out)=log(epsilon)
else
   pixel(im_out)=log(pixel(im_in))
```

L'image de sortie est de type Float.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Pour les graphes, le graphe de sortie est construit avec le logarithme des valeurs de noeuds.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *im_out*: une image de Floats ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule le logarithme de l'image tangram.pan :

```
plog tangram.pan a.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PLog( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

plogtransform

Transformations des niveaux de gris par loi logarithmique ou exponentielle.

Synopsis

plogtransform *inverse min max* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **plogtransform** étale ou compresse la plage des niveaux de gris de l'image d'entrée *im_in* en utilisant une transformation logarithmique ou exponentielle. Le paramètre *inverse* spécifie si la transformation est de type logarithmique ou exponentielle.

L'effet d'une transformation logarithmique est de plaquer une courte plage de niveaux de gris sombres sur une plus grande plage de niveaux de gris et de plaquer une grande plage de niveaux de gris clairs sur une courte plage de niveaux de gris en sortie. L'effet d'une transformation exponentielle est inverse.

La transformation logarithmique du pixel 'p' prend la forme :

```
im_out[p]=(c*log(im_in[p]-smin+1.0)) + min;
c=(max-min) / (log(smax-smin+1.0))
```

où smin et smax sont les valeurs the minimale et maximale de l'image d'entrée et c un facteur de normalisation pour l'étalement des valeurs de niveaux de gris en sortie entre *min* et *max*.

La transformation exponentielle d'un pixel 'p' prend la forme suivante :

```
im_out[p]=exp((im_in[p]-smin)/c) -1.0 + min;
c=(smax-smin) / (log(max-min+1.0))
```

où smin et smax sont les valeurs the minimale et maximale de l'image d'entrée et c un facteur de normalisation pour l'étalement des valeurs de niveaux de gris en sortie entre *min* et *max*.

Pour les images couleur et multispectrales, la transformation utilise l'approche vectorielle : le min et le max sont calculés sur toutes les bandes et chaque bande est modifiée avec la même transformation.

Paramètres

- *inverse* est un entier dans [0, 1] qui spécifie si la transformation est logarithmique (*inverse*=0) ou exponentielle (*inverse*=1).
- *min* et *max* spécifient les bornes des valeurs de pixel en sortie. Les valeurs possibles sont dépendantes du type de l'image d'entrée.
  Note: si *min* < *max* alors min et max sont affectés respectivement par les valeurs minimale et maximale possibles du type (eg. 0 and 255 pour les images Uchar).

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image avec les mêmes propriétés que l'image d'entrée *im_in*.

Résultat

Retourne SUCCESS ou FAILURE en cas de mauvais paramétrage.

Exemples

- Applique une transformation logarithmique suivie d'une transformation exponentielle pour créer l'image b.pan. Parce que la transformation exponentielle est l'inverse de la transformation logoarithmique, l'image de sortie est la même que l'image d'entrée (aux erreurs d'arrondi près). La différence des deux images doit être nulle:

```
plogtransform 0 0 255 tangram.pan a.pan
plogtransform 1 28 165 a.pan b.pan
```

- Applique une transformation logarithmique pour créer l'image a.pan en utilisant les bornes de sortie maximale du type de l'image:

```
plogtransform 0 1 -1 tangram.pan a.pan
```

Voir aussi

Transformation de la LUT, plineartransform, ppowerlawtransform

Prototype C++

```
Errc PLogTransform( const Img2duc &im_in, const Img2duc &im_out, int
inverse, float min, float max );
```

Auteur: Régis Clouard

pluv2lch

Changement d'espace couleur L*u*v vers LCH.

Synopsis

pluv2lch [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pluv2lch** de passer de l'espace couleur Luv à l'espace LCH (Light, Chroma, Hue) qui est une version perceptuelle de l'espace HSL.

Entrées

- *im_in*: une image couleur Luv.

Sorties

- *im_out*: une image couleur LCH.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image couleur a.pan de l'espace couleur Luv à LCH :

```
pluv2lch a.pan b.pan
```

Voir aussi

Color

Prototype C++

```
Errc PLUV2LCH( const Imc2duc &Ims, Imc2duc &im_out );
```

Auteur: Olivier Lezoray

pmalikperonafiltering

Lissage d'une image par diffusion non linéaire selon l'algorithme de Malik-Peronna.

Synopsis

pmalikperonafiltering *iterations hauteur* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pmalikperonafiltering** permet de lisser par diffusion non linéaire. L'algorithme utilisé est celui de Malik et Perona. Les dérivées sont calculées par des différences finies d'ordre 1.

Ce lissage permet de lisser les régions homogènes, tout en préservant les contours.

Paramètres

- *iterations* donne le nombre d'itération à opérer. Plus ce nombre est grand, plus le lissage est fort. Une valeur par défaut peut être 3.
- *hauteur* donne la hauteur de gradient des contours à préserver. Une valeur par défaut peut être 10.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applique un filtrage de Malik-Perona à l'image tangram.pan. Il préserve les contours d'amplitude > 10 et effectue 40 itérations :

```
pmalikperonafiltering 40 10 tangram.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PMalikPeronaFiltering( const Img2duc &im_in, Img2duc &im_out,
int iterations, int hauteur );
```

Auteur: Sophie Schüpp

pmanfr

Affichage en ligne de la documentation associée à un opérateur en français.

Synopsis

pmanfr [*-M path*] *nom_operateur ...*
pmanfr [*-M path*] *-k keyword ...*

Description

L'opérateur **pmanfr** permet d'afficher la documentation associée à un opérateur, à la manière du *man* d'Unix.

Dans le cas où le paramètre est le nom d'un opérateur, **pmanfr** affiche directement le contenu de la page html de l'opérateur concerné.

Dans le cas où le paramètre est un mot clé introduit par l'option *-k*), **pmanfr** affiche la liste des opérateurs qui référencent ce mot clé.

Paramètres

- *-k* permet de spécifier un mot clé à rechercher.
- *-M* permet de spécifier un répertoire où chercher la documentation. Le répertoire doit être organisé en doc/operatorsPxxx. Par exemple, la commande `pmanfr -M /usr/local/ pmanfr` permet de rechercher la documentation de l'opérateur `pmanfr` dans le répertoire `/usr/local/doc/operatorsP0`.

Résultat

Pas de valeur de retour.

Exemples

- Affiche le manuel de l'opérateur "pmanfr" :

```
pmanfr pmanfr
```

- Liste tous les opérateurs en relation avec le mot "segmentation" :

```
pmanfr -k segmentation
```

- Affiche le manuel de l'opérateur "pfoo" localisé dans le répertoire "/usr/local/myoperators/doc" :

```
pmanfr -M /usr/local/myoperators/doc pfoo
```

Voir aussi

Information

Auteur: Régis Clouard

pmask

Masquage d'un objet par une image ou une carte de régions.

Synopsis

pmask [*-m mask*] [*im_in1*|-][*im_in2*|-][*im_out*|-]

Description

L'opérateur **pmask** applique le masque *im_in2* sur l'image d'entrée *im_in1*.

Pour les images, le masquage est effectué sur chaque pixel:

```
if pixel(im_in2)
   pixel(im_out) = pixel(im_in1)
else
   pixel(im_out) = 0
```

Pour les images couleur ou multipsectrale, le masquage est effectué séparément sur chaque bande.

Pour les graphes, le masquage est appliqué sur les valeurs de noeud.

Pour les cartes de régions, le masquage est appliqué sur les labels. Tous les labels masqués sont mis à 0, les autres sont recopiés dans l'image de sortie.

Entrées

- *im_in1*: une image, un graphe ou une carte de régions.
- *im_in2*: une image de niveaux de gris ou une carte de régions.

Sorties

- *im_out*: un objet du même type que l'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Sélection des pixels des pièces de tangram :

```
pbinarization 100 1e30 examples/tangram.pan a.pan
pmask examples/tangram.pan a.pan b.pan
```

Voir aussi

Logique

Prototype C++

```
Errc PMask( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

pmassthresholding

Binarisation d'une image basé sur le pourcentage de niveaux de gris.

Synopsis

pcontrastthresholding *proportion* [*-m mask* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pmassthresholding** calcule la valeur de seuil à utiliser pour séparer l'image en deux classes telles que :

- la première contient *ratio*% des pixels avec une valeur < au seuil;
- la seconde, les 100-*ratio* pixels avec une valeur ≥ threshold.

Les pixels de valeur inférieure à la valeur correspondant à la proportion sont mis à 0, les autres sont mis à 255.

Paramètres

- *proportion* est un réel [0..100] correspondant au pourcentage souhaité pour la séparation.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *im_out*: une image de type char (Img2duc, Img3duc).

Résultat

Retourne la valeur de seuil.

Exemples

Supprime le fond d'image et garde les pièces de tangram:

```
pmassthresholding 86 tangram.pan out.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PMassThresholding( const Img2duc &im_in, Img2duc &im_out, Float
proportion );
```

Auteur: Régis Clouard

pmax

Maximum entre valeurs d'images ou de graphes.

Synopsis

pmax [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **max** calcule le maximum des valeurs de niveaux de gris entre chaque pixel de l'image *im_in1* et de l'image *im_in2* correspondant.

Le résultat est mis dans l'image destination *im_out* dont le type est celui du plus grand des deux images d'entrée *im_in1* ou *im_in2*

La formule de calcul est la suivante :

```
pixel(im_out) = maximum(pixel(im_in1),pixel(im_in2)).
```

Les deux images d'entrée *im_in1* ou *im_in2* doivent être de même type. Au besoin, il est nécessaire d'utiliser les opérateurs de coercition. L'image de sortie *im_out* est aussi du même type.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Pour les graphes, l'opérateur est appliqué sur les valeurs de noeuds.

Entrées

- *im_in1*: une image, un graphe ou une carte de régions
- *im_in2*: une image, un graphe ou une carte de régions

Sorties

- *im_out*: un objet du même type que les entrées.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pmax a.pan b.pan c.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PMax( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

pmaximumselection

Sélection de régions sur leur valeur de maximum intérieur.

Synopsis

pmaximumselection *relation seuil* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pmaximumselection** permet de sélectionner les régions sur leur valeur de maximum de niveau de gris intérieur calculé. dans *im_in*. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur entière correspondant à une valeur de niveaux de gris acceptée par le type de l'image.

Entrées

- *rg_in*: une carte de régions
- *im_in*: une image de niveaux de gris 2D ou 3D.

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Exemples

Sélectionne les régions avec un maximum > 50 :

```
pmaximumselection 1 50 rin.pan a.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PMaximumSelection( const Reg2d &rg_in, Img2duc &im_in, Reg2d
&rg_out, int relation, Uchar seuil );
```

Auteur: Régis Clouard

pmaximumvalue

Recherche de la valeur de pixel maximum dans une image (un graphe ou une carte de régions).

Synopsis

pmaximumvalue [*-m mask*] [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pmaximumvalue** retourne la valeur de pixel maximum dans *im_in*, ou de sommet dans un graphe ou du label pour une carte de régions.

Les valeurs maximales de chaque bande sont stockées dans la collection *col_out*.

Entrées

- *im_in*: une image ou une carte de régions ou un graphe.

Résultat

Retourne la valeur qui représente la valeur maximale de pixels dans l'image *im_in* (pour la première bande uniquement). Cette valeur est accessible par la commande **pstatus**.

Exemples

Mesure le maximum global de l'image tangram.pan (version Unix):

```
pmaximumvalue tangram.pan col.pan
val=‘pstatus‘
echo "Maximum = $val"
```

Mesure le maximum global de l'image tangram.pan (version MsDos):

```
pmaximumvalue tangram.pan col.pan
call pstatus
call pset val
echo Maximum = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Float PMaximumValue( const Img2duc &im_in, Collection & col_out );
```

Auteur: Régis Clouard

pmaxprojection

Projection orthogonale des valeurs maximales des pixels sur un axe d'une image.

Synopsis

pmaxprojection *axis* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pmaxprojection** consiste à construire une nouvelle image *im_out* de dimension inférieur à l'image d'entrée correspondant à la projection orthogonale d'un axe de l'image d'entrée. Ici c'est la valeur maximale dans la direction de la projection qui est reportée.

Par exemple, la projection d'une image 2D selon l'axe x construit une image 1D *im_out* de la largeur de l'image d'entrée où chaque pixel est affecté de la valeur :

$$im\_out[x] = MAX_y(im\_in[y][x])$$

Paramètres

- *axis* est un entier [0..3] qui indique l'axe de projection:
  - 0: en x,
  - 1: en y
  - 2: en z.

Entrées

- *im_in*: une image 2D ou 3D.

Sorties

- *im_in*: une image 1D ou 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Projection des niveaux de gris de l'image tangram.pan le long de l'axe des x :

```
pmaxprojection 0 tangram.pan a.pan
```

Voir aussi

Transformation

Prototype C++

```
Errc PMaxProjection( const Img3duc &im_in, Img2duc &im_out, int axis
);
```

Auteur: François Angot

pmcmfiltering

Lissage par diffusion par courbure moyenne.

Synopsis

pmcmfiltering *iterations* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pmcmfiltering** effectue une diffusion par courbure moyenne en 6-voisinage (Mean Curvature Motion).

C'est une approximation de la formule:

```
d(IMG/dt) = Curv(IMG).||grad(IMG)||
```

Paramètres

- *nombre_iterations* spécifie le nombre d'itération à effectuer l'opérateur.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Appliqye le filter lissage par courbure moyenne à l'image tangram.pan:

```
pmcmfiltering 2 tangram.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PMcmFiltering( const Img2duc &im_in, Img2duc &im_out, int
iterations );
```

Auteur: Sylvain Prudhon & Reynaud Joan

pmean

Moyennage entre images ou graphes.

Synopsis

pmean [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pmean** calcule l'image résultant de la moyenne entre les pixels de l'image *im_in1* et ceux de l'image *im_in2*.

La formule de calcul est la suivante :

```
pixel(im_out) = (pixel(im_in1) + pixel(im_in2))/2;
```

Les deux images d'entrés *im_in1* ou *im_in2* doivent être de même type. Au besoin, il est nécessaire d'utiliser les opérateurs de coercition. L'image de sortie est aussi de même type que les images d'entrée.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Pour les graphes, l'opérateur est appliquée sur la valeur des noeuds des graphes.

Entrées

- *im_in1*: une image ou un graphe.
- *im_in2*: une image ou un graphe.

Sorties

- *im_out*: un objet du même type que les entrées.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule le valeur moyenne entre les pixels des images a.pan et b.pan :

```
pmean a.pan b.pan c.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PMean( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

pmeanaggregation

Croissance des régions d'une carte selon la moyenne intérieure.

Synopsis

pmeanaggregation *connexite seuil* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pmeanaggregation** consiste à agglomérer des pixels à une région connexe lorsque sa valeur de pixel est proche de celle de la région (ie, l'écart avec la moyenne de la région est inférieur à la valeur de *seuil* donnée).

Les pixels à agglomérer sont les pixels non encore étiquétés dans la carte de régions *rg_in* (ceux qui ont un label=0).

La moyenne des régions de *rg_in* n'est pas recalculée pour éviter de trop s'éloigner de la situation initiale. On préférera des exécutions itératives de cet opérateur. On pourra par exemple itérer cet opérateur jusqu'à ce que le résultat de pstatus = 0. Ainsi, à chaque appel de l'opérateur la moyenne est recalculée avec les nouvelles régions.

La carte de sortie *rg_out* a le même nombre de labels que la carte d'entrée *rg_in*.

Paramètres

- *connexite* est lié à la dimension l'image 4 ou 8 pour le 2D, et 6 ou 26 pour le 3D.
- *seuil* fixe l'écart toléré à la moyenne d'une région pour y agglomérer un pixel.

Entrées

- *rg_in*: une carte de régions.
- *im_in*: une image de niveaux de gris.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre total de pixels qui ont été aggrégés à une région. Retourne FAILURE en cas de problème.

Exemples

Aggrège les pixels des pièces de tangram :

```
pbinarization 96 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pmeanaggregation 8 45 b.pan tangram.pan out.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PMeanAggregation( const Reg2d &rg_in, Img2duc &im_in, Reg2d
&rg_out, int connexite, Uchar seuil );
```

Auteur: Régis Clouard

pmeanfiltering

Lissage d'une image par un filtre moyenneur linéaire.

Synopsis

pmeanfiltering *halfsize* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pmeanfiltering** permet de lisser l'image d'entrée *im_in* par un filtre linéaire spatial. Il remplace un point par la moyenne de ses voisins. Par exemple, un filtre moyenneur 3x3 revient à remplacer un pixel par l'application du masque ci-dessous sur le pixel:

```
1/9  1/9  1/9
1/9  1/9  1/9
1/9  1/9  1/9
```

Chaque pixel est multiplier par 1/9 et la somme remplace le pixel central.

Le bord de l'image d'entrée (de taille halfsize) n'est pas traité, et se retrouve tel que dans l'image de sortie.

Paramètres

- Le paramètre *halfsize* permet de spécifier la demitaille du filtre >= 1. Une demitaille de 1 équivaut à un filtre de taille 3x3.
  Ce paramètre est inutilisé pour les graphes, mais doit être donné.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *im_out*: une image du même type que l'image d'entrée ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applique un filtre moyenneur de taille 5x5:

```
pmeanfiltering 2 tangram.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PMeanFiltering( const Img2duc &im_in, Img2duc &im_out, int
halfsize );
```

Auteur: Régis Clouard

pmeanmerging

Fusion prioritaire de régions selon la différence de moyennes intérieures.

Synopsis

pmeanmerging *nb_fusion seuil* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

L'opérateur **pmeanmerging** permet de fusionner les régions d'une carte de régions en utilisant le critère de la moyenne intérieure.

La notion de voisinage entre les régions est détenue par le graphe *gr_in*.

Le principe de l'algorithme est le suivant:
Pour chaque région de la carte de régions *im_in*, on calcule la différence de moyenne intérieure avec chacune de ses voisines. Si la différence est inférieure au *seuil* donné en paramètre, alors les régions sont fusionnées.
On utilise ici l'algorithme de croissance prioritaire qui consiste à fusionner à chaque fois les 2 régions dont la différence est la plus faible.

Paramètres

- *nb_fusion* permet de spécifier le nombre de fusion à effectuer (la valeur -1 signifie d'ignorer ce paramètre et donc d'exécuter l'algorithme tant qu'il y a des fusions possibles).
- *seuil* permet de spécifier la tolérance maximale sur l'écart des moyennes entre 2 régions. Les valeurs appartiennent à l'intervalle [0..nombre de niveaux de gris].

Entrées

- *rg_in*: une carte de régions.
- *gr_in*: une graphe.
- *im_in*: une image.

Sorties

- *rg_out*: une carte de régions.
- *gr_out*: un graphe.

Résultat

Retourne le nombre de fusions effectuées.

Exemples

Fusionne les régions issue d'une partition :

```
puniformityquadtree 0.9 tangram.pan a.pan
prg2gr a.pan b.pan
pmeanmerging -1 10 a.pan b.pan tangram.pan c.pan d.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PMeanMerging( const Reg2d &rg_in, const Graph2d &gr_in, const
Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, double nb_fusion,
Uchar seuil );
```

Auteur: Laurent Quesnel

pmeanprojection

Projection orthogonale sur un axe d'une image selon la somme.

Synopsis

pmeanprojection *axis* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pmeanprojection** consiste à construire une nouvelle image *im_out* de dimension inférieur à l'image d'entrée correspondant à la projection orthogonale d'un axe de l'image d'entrée. Ici c'est la valeur moyenne dans la direction de la projection qui est reportée.

Par exemple, la projection d'une image 2D selon l'axe x construit une image 1D *im_out* de la largeur de l'image d'entrée où chaque pixel est affecté de la valeur :

$$im\_out[x] = MOYENNE_y(im\_in[y][x])$$

Paramètres

- *axis* est un entier [0..3] qui indique l'axe de projection:
  - 0: en x,
  - 1: en y
  - 2: en z.

Entrées

- *im_in*: une image 2D ou 3D.

Sorties

- *im_out*: une image 1D ou 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Projection des niveaux de gris de l'image tangram.pan le long de l'axe des x :

```
pmeanprojection 0 tangram.pan a.pan
```

Voir aussi

Transformation

Prototype C++

```
Errc PMeanProjection( const Img3duc &im_in, Img2duc &im_out, int
axis );
```

Auteur: François Angot

pmeanselection

Sélection de régions sur leur valeur de moyenne intérieure.

Synopsis

pmeanselection *relation seuil* [*-m mask*] [*rg_in*|-] [*im_in*|-][*rg_out*|-]

Description

L'opérateur **pmeanselection** permet de sélectionner les régions sur leur valeur de moyenne intérieure calculée dans *im_in*. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

La moyenne mi de la région i est calculée par:

```
mi= SOMME(im_in[p] / p in Ri) / N
```

où N est le nombre de pixels de la region.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur entière correspondant à une valeur de niveau de gris acceptée par le type de l'image.

Entrées

- *rg_in*: une carte de régions
- *im_in*: une image de niveaux de gris 2D ou 3D.

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de régions ainsi sélectionnée.

Exemples

Sélectionne les régions avec une moyenne intérieure de = 50 :

```
pmeanselection 1 50 rin.pan a.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PMeanSelection( const Reg2d &rg_in, const Img2duc &im_in, Reg2d &rg_out, int relation, Uchar seuil );
```

Auteur: Régis Clouard

pmeanshiftsegmentation

Classification des pixels d'une image par l'algorithme Mean-Shift.

Synopsis

pmeanshiftsegmentation *spatial-bandwidth range-bandwidth minimum-region-area speed-up-level* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pmeanshiftsegmentation** construit une carte de régions en classifiant les pixels selon leur homogénéité en couleur et leur proximité spatiale. C'est une procédure pour localiser les maxima d'une fonction densité donnée à partir de données discrètes. L'algorithme est basé sur la détection des modes dans la fonction densité donnée.

Les paramètres permettent de contrôler le processus de classification. Ainsi, *spatial-bandwidth* contrôle la taille de la fenêtre spatiale de recherche et *range-bandwidth* contrôle la largeur de la bande de la fenêtre de recherche spectrale. Plus les fenêtres sont grandes, moins il y aura de régions en sortie.

Le résultat est une carte de régions *rg_out*.

Paramètres

- *spatial-bandwidth* spécifie la taille de la fenêtre de recherche des modes: (2r+1)*(2r+1), où *r* est la valeur de *spatial-bandwidth*. C'est un entier avec une valeur strictement supérieure à 0, mais qui ne doit pas être trop grande pour ne pas ralentir le temps de traitement (généralement <10).
- *range-bandwidth* spécifie la largeur de bande pour la recherche. C'est une valeur réelle supérieure à zero.
- *minimum-region-area* spécifie la surface minimale acceptable pour une région du résultat. C'est un entier supérieur à 0 donné en pixels.
- *speedup-level* détermine le niveau de rapidité de calcul. C'est une valeur entière dans [0..2], où 2 est la valeur maximale. Cependant, la précision des résultats; se fait au détriment de la vitessse : 0 donne donc les meilleurs résultats. Il faut noter que la taille du paramètre *spatial-bandwidth* influe aussi sur la vitesse.

Entrées

- *im_out*: une image 2D.

Sorties

- *im_out*: une carte de régions.

Résultat

Retourne le nombre de régions formées ou FAILURE.

Exemples

Segmente les pièces de tangram:

```
pmeanshiftsegmentation 7 6.5 100 0 examples/tangram.pan a.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PMeanShiftSegmentation( const Imc2duc &ims, Reg2d &rgd, int
spatialBandwidth, float rangeBandwidth, int minimumRegionArea, int
speedup ); int halfsize );
```

Référence

D. Comanicu, P. Meer: "Mean shift: A robust approach toward feature space analysis." IEEE Trans. Pattern Anal. Machine Intell., 24, 603-619, May 2002

Auteur: Régis Clouard

pmeanvalue

Calcul du niveau de gris moyen d'une image.

Synopsis

pmeanvalue [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pmeanvalue** retourne la valeur moyenne des valeurs pour les *pixels non nuls* de l'image *im_in*, ou des sommets s'il s'agit d'un graphe.

La moyenne est faite selon la formule:

```
moyenne = SOM (im_in(x,y)) / N; si im-In(x,y) != 0.
```

où N est le nombre de pixels (ou noeuds).

Les valeurs moyennes de chaque bande sont stockées dans la collection *col_out*.

Remarque : Cet opérateur n'est pas masquable.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *col_out*: une collection de valeurs réelles.

Résultat

Retourne la valeur réelle qui représente la moyenne totale de l'image *im_in*. Cette valeur est accessible par la commande **pstatus**.

Exemples

Mesure la moyenne globale de l'image tangram.pan (version Unix):

```
pmeanvalue tangram.pan col.pan
var=‘pstatus‘
echo "Moyenne = $val"
```

Mesure la moyenne globale de l’image tangram.pan (version MsDos):

```
pmeanvalue tangram.pan col.pan
cal pstatus
cal pset var
echo Moyenne = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Float PMeanValue( const Img2duc &im_in, Collection & col_out );
```

Auteur: Régis Clouard

pmedianfiltering

Lissage d'une image par médian standard séparable.

Synopsis

pmedianfiltering *demitaille* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pmedianfiltering** permet d'appliquer un filtre de lissage non linéaire de type médian. Il consiste simplement à remplacer un pixel par la valeur médiane de ses voisins définie sur la fenêtre de taille (demitaille*2) centrée sur le pixel.

C'est un filtre adapté au lissage d'image à bruit impulsionnel ou exponentiel. Il enlève les petits détails tout en préservant les contours de type "marche" et l'image de sortie ne contient aucune nouvelle intensité. Par contre, ce filtre peut affecter la géométrie des régions de l'image, par exemple les zones présentant un angle aigu ont tendance à être arrondie. Il a aussi tendance à faire disparaître les contours de type "arête" et "toit".

Paramètres

- Le paramètre *demitaille* spécifie la demitaille du filtre [1..n] pour le calcul de la valeur mediane. Plus la valeur du filtre est grande plus le lissage est fort.

Entrées

- *im_in*: une image de niveaux de gris.

Entrées

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applique un filtre médian sur l'image tangram.pan:

```
pmedianfiltering 2 tangram.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PMedianFiltering( const Img2duc &im_in, Img2duc &im_out, int
demitaille );
```

Auteur: Julien Robiaille

pmedianvalue

Recherche de la valeur médiane d'une image.

Synopsis

pmedianvalue [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pmedianvalue** permet de retourner la valeur médiane des valeurs de pixel d'une image.

Si le nombre de pixels est impair alors la valeur médiane est celle au milieu de la liste des valeurs de pixels triées par ordre croissant, et si le nombre de pixels est pair alors la valeur est la moyenne entre les deux valeurs autour du centre.

Les valeurs médianes de chaque bande sont stockées dans la collection *col_out*.

Entrées

- *im_in* : une image ou un graphe.

Sorties

- *col_out*: une collection de valeurs réelles.

Résultat

Retourne la valeur réelle qui représente la valeur médiane dans l'image *im_in* (pour la première bande uniquement). Cette valeur est accessible par la commande **pstatus**.

Exemples

Mesure la valeur médiane de l'image tangram.pan (version Unix):

```
pmedianvalue tangram.pan col.pan
var=‘pstatus‘
echo "Mediane = $val"
```

Mesure la valeur médiane de l'image tangram.pan (version MsDos):

```
pmedianvalue tangram.pan col.pan
call pstatus
call pset var
echo Mediane = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Float PMedianeValue( const Img2dsf &im_in, Collection & col_out );
```

Auteur: Jalal Fadili

pmergeimages

Regroupement de 4 sous-images en une seule.

Synopsis

pmergeimages [*im_in1*|-] [*im_in2*|-] [*im_in3*|-] [*im_in4*|-] [*im_out*| -]

Description

L'opérateur **pmergeimages** permet de créer une seule image de dimensions (2M)x(2N) à partir de 4 images (0, 1, 2, 3) de mêmes dimensions MxN suivant ce schéma:

```
[0][1]
[2][3]
```

Les images d'entrée sont supposées être de même type. Les tailles doivent être compatibles:

- *im_in1* colonnes = *im_in3* colonnes;
- *im_in2* colonnes = *im_in4* colonnes;
- *im_in1* lignes = *im_in2* lignes;
- *im_in3* lignes = *im_in4* lignes;

La taille de l'image de sortie est (*im_in1* + *im_in2*) de large et (*im_in1* + *im_in3*) de haut.

Entrées

- *im_in1*, *im_in2*, *im_in3*, *im_in4*: des images 2D de même type et de tailles compatibles.

Entrées

- *im_out*: une image 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit une image pour illustrer le phénomène de Gibbs lors d'une analyse par ondelette.

```
pshapedesign 256 256 0 2 150 150 a.pan
pqmf daubechies 4 b.pan
pdwt 1 a.pan b.pan c.pan
psplitimage c.pan d1.pan d2.pan d3.pan d4.pan
pthresholding 20 400 d2.pan e2.pan
pthresholding 20 400 d3.pan e3.pan
pthresholding 20 400 d4.pan e4.pan
pmergeimages d1.pan e2.pan e3.pan e4.pan f.pan
pidwt 1 f.pan b.pan out.pan
```

Voir aussi

Utilitaire, psplitimage

Prototype C++

```
Errc PMergeImages( const Img2dsd &im_in1, const Img2dsd &im_in2,
const Img2dsd &im_in3, const Img2dsd &im_in4, Img2dsd &im_out );
```

Auteur: Ludovic Soltys

pmin

Minimum entre valeurs d'images ou de graphes.

Synopsis

pmin [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pmin** calcule le minimum des valeurs de niveaux de gris entre chaque pixel de l'image *im_in1* et de l'image *im_in2* correspondant.

Le résultat est mis dans l'image destination *im_out* dont le type est celui du plus grand des deux images d'entrée *im_in1* ou *im_in2*

La formule de calcul est la suivante :

```
pixel(im_out) = minimum(pixel(im_in1),pixel(im_in2))
```

Les deux images d'entrés *im_in1* ou *im_in2* doivent être de même type. Au besoin, il est nécessaire d'utiliser les opérateurs de coercition. L'image de sortie *im_out* est aussi du même type.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Pour les graphes, l'opérateur est appliqué sur les valeurs de noeuds.

Entrées

- *im_in1*: une image ou un graphe.
- *im_in2*: une image ou un graphe.

Sorties

- *im_out*: un objet du même type que les entrées.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule le minimum entre les images a.pan et b.pan :

```
pmin a.pan b.pan c.pan
```

Voir aussi

Arithmetiue

Prototype C++

```
Errc PMin( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

pminimumselection

Sélection de régions sur leur valeur de minimum intérieur.

Synopsis

pminimumselection *relation seuil* [*-m mask*] [*rg_in*|-] [*im_in*|-][*rg_out*|-]

Description

L'opérateur **pminimumselection** permet de sélectionner les régions sur leur valeur de minimum de niveau de gris intérieur calculée dans *im_in*. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur entière correspondant à une valeur de niveau de gris acceptée par le type de l'image.

Entrées

- *rg_in*: une carte de régions
- *im_in*: une image de niveaux de gris 2D.

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Exemples

Sélectionne les régions avec la plus petite valeur de nievaxu de gris :

```
pminimumselection -3 0 rin.pan a.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PMinimumSelection( const Reg2d &rg_in, Img2duc &im_in, Reg2d
&rg_out, int relation, Uchar seuil );
```

Auteur: Régis Clouard

pminimumvalue

Recherche de la valeur de pixel minimum dans une image (un graphe ou une carte de régions).

Synopsis

pminimumvalue [*-m mask*] [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pminimumvalue** retourne la valeur de pixel minimum dans *im_in*, ou de sommet dans un graphe ou du label pour une carte de régions.

Les valeurs minimales de chaque bande sont stockées dans la collection *col_out*.

Entrées

- *im_in*: une image ou une carte de régions ou un graphe.

Sorties

- *col_out*: une collection de valeurs réelles.

Sorties

- *col_out*: une collection de valeurs réelles.

Résultat

Retourne la valeur qui représente le nombre minimum des pixels dans l'image *im_in* (pour la première bande uniquement). Cette valeur est accessible par la commande **pstatus**.

Exemples

Mesure le minimum globale de l'image tangram.pan (version Unix):

```
pminimumvalue tangram.pan col.pan
var=‘pstatus‘
echo "Minimum = $val"
```

Mesure le minimum globale de l'image tangram.pan (version MsDos):

```
pminimumvalue tangram.pan col.pan
call pstatus
call pset var
echo Minimum = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Float PMinimumValue( const Img2duc &im_in, Collection & col_out );
```

Auteur: Régis Clouard

pmitchellrescale

Retaille d'une image par l'algorithme de Mitchell.

Synopsis

pmitchellrescale *rescalex rescaley rescaleyz* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pmitchellrescale** utilise un noyau de convolution pour interpoler les valeurs des pixels de l'image d'entrée *im_in* afin de calculer les valeurs des pixels de l'image de sortie *im_out*.
L'interpolation consiste à pondérer l'influence des pixels d'entrée. Les poids sont dépendants de la position du pixel de sortie et sont donnés par l'algorithme de Mitchell:

```
Soit tt=sqr(x), B=1/3, C=1/3

       | (((12-9*B-6*C)*(x*tt)) + ((-18+12*B+6*C)*tt) + (6-2*B)) / 6 si -1<x<1
M(x) = | (((-1*B-6*C) * (x*tt)) + ((6*B+30*C) *tt) + ((-12*B-48*C)*x) + (8*B+24*C)) / 6 si -2<x<2
       | 0 sinon
```

Par exemple, si l'image est zoomée par 3, alors chaque pixel de sortie est donné par:

```
for i in [-2, 2]
  for j in [-2, 2]
      im_out[p.y][p.x] += M(i*scalex)*M(j*scaley)*im_in[p.y*scaley+j][p.x*scalex+i]
```

Pour zoomer une carte de régions ou un graphe, il faut utiliser l'operateur prescale.

Paramètres

- *rescalex*, *rescaley*, *rescalez* sont des réels positifs correspondant aux facteurs de retaille.
  Si les rescales sont > 1 alors il s'agit d'un agrandissement, s'ils sont < 1 alors il s'agit d'une réduction.
  rescalez est ignoré pour le cas des images 2D mais doit être donné.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Agrandissement de l’image d’un facteur 2 :

```
pmitchellrescale 2 2 0 tangram.pan a.pan
```

Réduction de l’image d’un facteur 2 :

```
pmitchellrescale 0.5 0.5 0 tangram.pan a.pan
```

Voir aussi

Transformation, plinearrescale, pbicubicrescale, pbellrescale, phermiterescale, planczosrescale, pquadraticbsplinerescale, prescale

Prototype C++

```
Errc PMitchellRescale( const Img2duc &im_in, Img2duc &im_out, float rescaley, float rescalex );
```

Auteur: Régis Clouard

pmodevalue

Recherche de la valeur de pixel la plus nombreuse dans l'image (un graphe ou une carte de régions).

Synopsis

pmodevalue [*-m mask*] [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pmodevalue** retourne la valeur de pixel la plus nombreuse dans l'image *im_in*, ou de sommet dans un graphe ou du label pour une carte de régions.

Les modes de chaque bande sont stockées dans la collection *col_out*.

Entrées

- *im_in*: une image ou une carte de régions ou un graphe.

Résultat

Retourne la valeur qui est la plus fréquente dans l'image *im_in* (pour la première bande uniquement). Cette valeur est accessible par la commande **pstatus**.

Exemples

Retourne le mode dans l'image tangram.pan (version Unix):

```
pmodevalue tangram.pan col.pan
val=‘pstatus‘
echo "Mode = $val"
```

retourne le mode de l'image tangram.pan (version MsDos):

```
pmodevalue tangram.pan col.pan
call pstatus
call pset val
echo Mode = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Float PModeValue( const Img2duc &im_in, Collection & col_out );
```

Auteur: Régis Clouard

pmodulus

Calcul du module entre deux images.

Synopsis

pmodulus [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pmodulus** calcule le module des valeurs de niveaux de gris entre chaque pixel de l'image *im_in1* et de l'image *im_in2* correspondant. Le résultat est mis dans l'image destination *im_out* qui est de type réel.

Cet opérateur est notamment utilisé pour calculer le module d'une image complexe (partie réelle et partie imaginaire).

La formule de calcul est la suivante :

```
pixel(im_out) = sqrt(pixel(im_in1)*pixel(im_in1) + pixel(im_in2)*pixel(im_in2))
```

Entrées

- *im_in1*: une image de niveaux de gris ou de couleur.
- *im_in2*: une image de niveaux de gris ou de couleur.

Sorties

- *im_out*: une image de Float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit une image de carré du domaine spatial dans le domaine fréquentiel et réciproquement :

```
pshapedesign 256 256 0 2 20 0 square.pan
pshapedesign 256 256 0 0  0 0 empty.pan
pfft square.pan empty.pan real.pan imaginary.pan
pmodulus real.pan imaginary.pan modulus.pan
pphase real.pan imaginary.pan phase.pan
pifft real.pan imaginary.pan square1.pan empty1.pan
plineartransform 0 0 255 square1.pan square2.pan
pim2uc square2.pan newsquare.pan
```

Voir aussi

Domaine Fréquentiel, pphase

Prototype C++

```
Errc PModulus( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

pmse

Calcul de l'Erreur Quadique Moyenne (Mean Square Error).

Synopsis

pmse [*im_in1*|-] [*im_in2*|-]

Description

L'opérateur **pmse** mesure l'erreur quadratique moyenne (MSE Mean Squared Error) entre l'image initiale *im_in1* et sa version restaurée ou améliorée *im_in2*.
Une valeur faible signifie une faible erreur. Cepndant, le MSE dépend de la valeur de pixel maximale de l'image d'entrée. Par exemple, un MSE=100.0 pour image de Uchar image est très élevé alors qu'il est faible pour une image de Long.

MSE est défini comme suit :

```
MSE=1/N * sum { (im_in1(pixel)-im_in2(pixel))^2 }
```

où N est le nombre total de pixel de l'image d'entrée *im_in1*.

Les images d'entrée *im_in1* et *im_in2* doivent avoir le même type et les mêmes dimensions.

Pour les images couleur et multispectrales, la définition de MSE est la même excepté que c'est la somme de toutes les différences des carrés divisée par la taille de l'image et le nombre de bandes qui est utilisé.

Entrées

- *im_in1*: une image.
- *im_in2*: une image (une version restaurée ou ameliorée version de im_in1).

Résultat

Retourne une valeur réelle positive.
(Utiliser `pstatus` pour récupérer cette valeur).

Exemples

Calcule le MSE pour le filtre moyenneur:

```
pmeanfilter 2 tangram.pan i1.pan
pmse tangram.pan i1.pan
pstatus
```

Voir Aussi

Evaluation, psnr, ppsnr

Prototype C++

```
Errc PMSE( const Img2duc &im_in1, const Img2duc &im_in2 );
```

Auteur: Régis Clouard

pmst

Construction de l'arbre de recouvrement minimal d'un graphe.

Synopsis

pmst [*-m mask*] [*gr_in*|-] [*gr_out*|-]

Description

L'opérateur **pmst** permet de calculer l'arbre de recouvrement minimal du graphe *gr_in*.

L'arbre de recouvrement minimal est l'arbre de recouvrement dont la somme des poids des arcs est minimum. Cette structure moins dense que le graphe d'origine ne contient que les arcs entre les sommets les plus proches au sens d'une distance quelconque (ici euclidienne) et donnée dans le poids des arcs.

La technique utilisée correspond à l'algorithme de Prim. Cet algorithme est basé sur le grossissement d'un sous-graphe jusqu'à recouvrement en choisissant d'ajouter à chaque fois l'arête de plus faible coût qui ne crée pas un cycle. On prend comme racine de l'arbre le dernier sommet du graphe (pure convention).

Les liens entre les sommets sont modifiés de manière physique; les relations de voisinage initiales sont perdues et les poids des arcs restants sont mis à 1.

Entrées

- *gr_in*: un graphe.

Sorties

- *gr_out*: un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pmst g1.pan g2.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PMst( const Graph2d &gr_in, Graph2d &gr_out );
```

Auteur: François Angot

pmult

Multiplication d'images ou de graphes.

Synopsis

pmult [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pmult** affectue la multiplication de deux images de n'importe quel type. L'opération est faite entre chaque pixel de l'image *im_in1* et de l'image *im_in2* correspondant.

Il n'y a pas de gestion du débordement de valeurs. La formule reprend exactement l'opérateur du C :

```
pixel(im_out) = (pixel(im_in1) * pixel(im_in2));
```

Les deux images d'entrée *im_in1* ou *im_in2* doivent être de même type. Au besoin, il est nécessaire d'utiliser les opérateurs de coercition. Par contre, l'image de sortie est du type le plus grand possible par rapport au type des images d'entrée:

- Long entre images d'octets.
- Long entre images d'entiers.
- Float entre image de floats.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Pour les graphes, l'opérateur est appliqué sur les noeuds du graphe.

Entrées

- *im_in1*: une image ou un graphe.
- *im_in2*: une image ou un graphe.

Sorties

- *im_out*: une image ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pmult a.pan b.pan c.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PMult( const Img2duc &im_in1, const Img2duc &im_in2, Img2dsf
&im_out );
```

Auteur: Régis Clouard

pmultcst

Multiplication d'une image, d'un graphe ou d'une carte de région par une constante.

Synopsis

pmultcst *cst* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pmultcst** calcule l'image *im_out* par multiplication des valeurs de pixels de l'image *im_in* par la valeur *cst*.

Dans tous les cas, l'image de sortie est du même type que l'image d'entrée.

Il y a écrétage du résultat si la valeur résultante est supérieure à la valeur maximale du type de l'image.

La formule de calcul est la suivante :

```
pixel(im_out) = pixel(im_in) * cst;
```

Pour une carte de régions, ce sont les valeurs des labels qui sont multipliées.

Pour un graphe, ce sont les valeurs des noeuds qui sont multipliées.

Paramètres

- *cst* est une valeur réelle.

Entrées

- *im_in*: une image, un graphe ou une carte de régions.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Pour les cartes de régions, retourne la valeur de label maximale.

Exemples

Division des pixels de l’image tangram.pan par 2 :

```
pmultcst 0.5 tangram.pan a.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PMultCst( const Img2duc &im_in, Img2duc &im_out, float cst );
```

Auteur: Régis Clouard

pmultval

Multiplication d'une image par des constantes stockées dans une collection.

Synopsis

pmultval [*-m mask*] [*col_in*|-] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pmultval** calcule l'image *im_out* par multiplication des valeurs de pixels de l'image *im_in* par les valeur stockes dans la collection *col_in*. La première valeur de la collection est utilisée pour multiplier tous les pixels de la première bande, la seconde à tous les pixels de la seconde bande, etc.

Il y a écrétage du résultat si la valeur résultante est supérieure à la valeur maximale du type de l'image. La formule de calcul est la suivante :

```
val = pixel(im_in) * col_in;
if (val > MAX) pixel(im_out) = MAX;
else if (val < MIN) pixel(im_out) = MIN;
else pixel(im_out) = val;
```

Entrées

- *col_in*: une collection avec autant de valeurs réelles que de nombre de bandes pour l'image d'entrée (p. ex. 3 pour une image couleur).
- *im_in*: une image.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Multiplie tangram.pan sa valeur moyenne:

```
pmeanvalue tangram.pan a.pan
pmultval a.pan tangram.pan b.pan
```

Autres exemples

Voir aussi

Arithmetique

Prototype C++

```
Errc PMultVal( const Collection &col_in, const Img2duc &im_in,
Img2duc &im_out );
```

Auteur: Régis Clouard

pmumfordshahmerging

Fusion prioritaire de régions selon la variation d'énergie de Mumford Shah.

Synopsis

pmumfordshahmerging *nb_fusion alpha seuil* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

L'opérateur **pmumfordshahmerging** permet de fusionner les régions d'une carte de régions en utilisant le critère de la variation d'énergie de Mumford Shah.

La notion de voisinage entre les régions est détenue par le graphe *gr_in*.

Le principe de l'algorithme est le suivant:
Pour chaque région de la carte de régions *rg_in*, on calcule la variation d'énergie entre une région et ses voisines. Si la différence est inférieure au *seuil* donné en paramètre, alors les régions sont fusionnées.
On utilise ici l'algorithme de croissance prioritaire qui consiste à fusionner à chaque fois les 2 régions dont la différence est la plus faible.

On calcule la variation d'énergie (DE) de Mumford Shah par la formule suivante :

$$DE = \frac{(Card(R1)+Card(R2))*(moy(R1)-moy(R2))^2 - 2*alpha*frontiere(R1,R2)}{Card(R1)*Card(R2)}$$

où: alpha est un paramètre
moy(R1) est la moyenne des niveaux de gris d'une région
frontiere(R1,R2) est la longueur de la frontière entre R1 et R2
Card(R1) est le nombre de pixels de la région

Les valeurs négatives pour delta_energie signifient que l'énergie de l'union des deux régions est inférieure à la somme des énergies des deux régions. Dans ce cas, on peut fusionner les deux régions.

Paramètres

- *nb_fusion* permet de spécifier le nombre de fusion à effectuer (la valeur -1 signifie d'ignorer ce paramètre et donc d'exécuter l'algorithme tant qu'il y a des fusions possibles).

- *alpha* permet de spécifier l'importance que l'on donne aux longueurs de frontières. Les valeurs sont positives et peuvent atteindre 2000.
- *seuil* permet de spécifier la tolérance maximale sur la variance de l'énergie à prendre en compte. Ce nombre peut-être négatif (généralement il vaut 0).

Entrées

- *rg_in*: une carte de régions.
- *gr_in*: une graphe.
- *im_in*: une image.

Sorties

- *rg_out*: une carte de régions.
- *gr_out*: un graphe.

Résultat

Retourne le nombre de fusions effectuées.

Exemples

Fusionne les régions issues d'une partition :

```
puniformityquadtree 0.9 tangram.pan a.pan
prg2gr a.pan b.pan
pmumfordshahmerging -1 5 1  a.pan b.pan tangram.pan c.pan d.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PMumfordshahMerging( const Reg2d &rg_in, const Graph2d &gr_in,
const Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, Long
nb_fusion, double alpha, double seuil );
```

Auteur: Laurent Quesnel

pnagaofiltering

Lissage par maximum d'homogénéité selon le masque de Nagao.

Synopsis

pnagaofiltering [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pnagaofiltering** effectue un lisage de l'image d'entrée. Le lissage de Nagao procède par fractionnement du voisinage en domaines distincts, pour lesquels le critère d'homogénéité de la variance est calculé. Le domaine le plus homogène (ie, celui ayant la variance la plus faible) est sélectionnée. le point central est remplacé par la valeur moyenne de ce domaine. Le filtre de Nagao utilise 9 domaines de taille 5x5:

```
La matrice de base et ses 8 autres rotations.
   rotation 0:
     |0,1,1,1,0|
     |0,1,1,1,0|
     |0,0,1,0,0|
     |0,0,0,0,0|
     |0,0,0,0,0|

   rotation 1:
     |0,0,0,1,1|
     |0,0,0,1,1|
     |0,0,1,1,1|
     |0,0,0,0,0|
     |0,0,0,0,0| etc.
```

Cet opérateur présente l'avantage de réduire le bruit tout en renforcant les contrastes par diminution de la largeur de bande de la zone de transition.

Entrées

- *im_in*: une image 2D de niveaux de gris.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applique le lissage de Nagao à l'image tangram.pan:

```
pnagaofiltering tangram.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PNagaoFiltering( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

pnewcollection

Création d'une collection vide.

Synopsis

pnewcollection [*col_out*|-]

Description

L'opérateur **pnewcollection** retourne une nouvelle collection vide.

Sorties

- *im_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Concaténation de 2 collections :

```
pnewcollection col.pan
pobject2col bar parrot.pan c1.pan
pcolcatenateitem col.pan c1.pan col.pan
pobject2col bar tangram.pan c1.pan
pcolcatenateitem col.pan c1.pan col.pan
```

Voir aussi

Collection.

Prototype C++

Auteur: Régis Clouard

pnewimage

Création d'une nouvelle image.

Synopsis

pnewimage *w h d val* [*im_out*|-]

Description

L'opérateur **pnewimage** crée une nouvelle image de taille (*w: largeur,h: hauteur,d: profondeur*) avec la valeur *val* pour tous les pixels.

Si *h*<=0 alors l'image de sortie est une image 1D
Si *d*<=0 alors l'image de sortie est une image 2D
sinon une image 3D.

Le type de l'image de sortie dépend de la valeur de *val*. Si *val* est un entier < 255 et ≥ 0 alors l'image de sortie est une image d'octets; si *val* est un entier > 255 ou < 0 alors l'image de sortie est une image de longs; si *val* est un réel l'image de sortie est une image de réels.

Deux autres opérators peuvent être utilisé pour créer une nouvelle image. psetcst crée une nouvelle image à partir des propriétés d'une autre image. pshapedesign crée une nouvelle image à partir de la spécification de dimensions et d'un type.

Paramètres

- *w,h,d* (largeur, hauteur, profondeur) spécifient la taille de l'image de sortie.
- *val* donne la valeur pour tous les pixels de l'image. Ce peut être un entier court ou long ou un réel.

Sorties

- *im_out*: une image dont le type dépend de *val*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Comble les trous dans les régions obtenues par une segmentation de l'image tangram.pan.

```
pbinarization 100 1e30 tangram.pan in.pan
pnewimage 256 256 0 255 i0.pan
psetborder 1 1 1 1 1 1 0 i0.pan i1.pan
perosionreconstruction 4 i1.pan in.pan fillhole_out.pan
```

Voir aussi

Utilitaire

Prototype C++

```
Pas de prototype.
```

Auteur: Régis Clouard

pniblackbinarization

Binarisation de l'image basée sur le contraste local selon la méthode de W. Niblack améliorée par J. Sauvola.

Synopsis

pniblackbinarization *width height depth k* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pniblackbinarization** est une binarisation adaptative qui opère sur une fenêtre glissante. Il classifie les pixels de l'image d'entrée *im_in* en 2 classes : fond et objets. L'algorithme est basé sur le calcul du contraste local pour chaque pixel qui dépend du contraste moyen et de son écart-type calculés dans un voisinage autour du pixel.

Le principe de l'algorithme est de glisser une fenêtre carrée sur l'image. La fenêtre définit la taille du voisinage. Elle doit être suffisamment petite pour préserver le contraste local et suffisament grande pour supprimer le bruit. Le seuil T pour le pixel central de la fenêtre est calculé en utilisant la moyenne *m* et l'écart-type *s*:

```
T = m . (1 - k. ( 1- s/R))
```

où R est la dynamique de l'écart-type (p.ex., 128 avec les images 8-bits) et *k* une constante avec des valeurs positives (p.ex., 0.5).

Remarque: cet algorithme utilise l'hypothèse forte que les objets sont sombres (proche de 0) et le fond est clair (proche de 255).

Paramètres

- *width height, depth* fixent la taille de la fenêtre pour calculer le contraste.
- *k* détermine combien les frontières des objets influent sur l'identification des objets. La valeur par défaut est 0.5.

Entrées

- *im_in*: une image de niveaux de gris (Img2duc, Img3duc).

Sorties

- *im_out*: une image de niveaux de gris (Img2duc, Img3duc).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Segmente l'image tangram.pan en 2 classes ; le fond et les pièces:

```
pinverse examples/tangram.pan a.pan
pniblackbinarization 50 50 0 0.2 a.pan out.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PNiblackBinarization( const Img2duc &im_in, Img2duc &im_out,
int width, int height, int depth, float k );
```

Références

W. Niblack, "*An introduction to digital image processing*", Prentice hall, pp. 115-116, 1986.

J. Sauvola, M. Pietikainen, "Adaptative document ipage binarization", *Pattern Recognition*, vol 33, pp 255-236, 2000.

Auteur: Régis Clouard

pnodedisc

Visualisation des valeurs des noeuds d’un graphe.

Synopsis

pnodedisc [*gr_in*|-] [*rg_out*|-]

Description

L’opérateur **pnodedisc** permet de visualiser la valeur des sommets d’un graphe en dessinant aux coordonnées de chaque sommet un disque de rayon égal à sa "valeur".

La carte de régions de sortie *rg_out* est une carte de régions dans laquelle l’étiquette attribuée è chaque disque est égale à la valeur du rang du sommet dans la liste des sommets.

Entrées

- *gr_in*: un graphe.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de régions correspondantes.

Exemples

```
pnodedisc g1.pan r.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PNodeDisc( const Graph &gr_in, Reg2d &rg_out );
```

Auteur: François Angot

pnodevisu

Visualisation des valeurs des sommets d'un graphe dans une image.

Synopsis

pnodevisu [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_out*|-]

Description

L'opérateur **pnodevisu** permet de visualiser la valeur attribuée à chacun des sommets du graphe *gr_in*.

Chacune des régions de *rg_in* est colorée avec la valeur du champ "value" du sommet correspondant dans *gr_in*.

Entrées

- *rg_in*: une carte de régions
- *gr_in*: un graphe

Sorties

- *im_out* : une image.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pnodevisu r.pan g.pan i.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PNodeVisu( const Reg2d &rg_in, const Graph2d &gr_in, Img2dsl
&im_out );
```

Auteur: François Angot

pnonlocaldilation

Dilatation par régularization non locale du laplacien.

Synopsis

pnonlocaldilation sigma number_of_iterations connectivity [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pnonlocalerosion** applique une érosion non locale sur l'image *im_in*.
Soit f l'image d'entrée *im_in*, l'algorithme s'applique itérativement *number_of_iteration* fois :
$ f(u)^{t+1} = f^t(u) + \max_{v ~ u}(w(u, v) \max(f^t(v) - f^t(u), 0))$
avec u un pixel, v un voisin de u, $w(u, v)$ le poids entre u et v (qui peut être une mesure de similarité, une distance, etc).

Ici nous utilisons une mesure de similarité exponentielle: $w(u,v)=\exp{-distance(f(u), f(v))^2 / sigma^2}$. Avec $w(u, v) = 1$, on obtient une dilatation classique.

Paramètres

- *sigma*: seuil maximal de la mesure de similarité.
- *number_of_iterations*: Le nombre d'itérations.
- *connectivity*: la connexité: 4 ou 8.

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Applique une dilatation non locale à l'image tangram.pan, avec une 8-connectivité, 10 iterations, et sigma = 15:

```
pnonlocaldilation 15 10 8 tangram.pan out.pan
```

Voir aussi

pnonlocalerosion

Prototype C++

```
Errc PNonLocalDilation( const Img2duc & ims, Img2duc & imd, float
sigma, int nbIter, int ngbIdx );
```

Auteur: Matthieu Toutain

pnonlocalerosion

Erosion par régularization non locale du laplacien.

Synopsis

pnonlocalerosion sigma number_of_iterations connectivity [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pnonlocalerosion** applique une érosion non locale sur l'image *im_in*.
Soit f l'image *im_in*, l'algorithme s'applique itérativement:
$ f(u)^{t+1} = f^t(u) + \min_{v ~ u}(w(u, v) \min(f^t(v) - f^t(u), 0))$
avec u un pixel, v un voisin de u, $w(u, v)$ le poids entre u aet v (qui peut être une mesure de similarité, une distance, etc).

Ici nous utilisons une mesure de similarité exponentielle: $w(u,v) = \exp{-distance(f(u), f(v))^2 / sigma^2}$. Avec $w(u, v) = 1$, on obtient une érosion classique.

Paramètres

- *sigma*: seuil maximal de la mesure de similarité.
- *number_of_iterations*: Le nombre d'itérations.
- *connectivity*: la connexité: 4 ou 8.

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Applique une érosion non locale à l'image tangram.pan, avec une 8-connectivité, 10 iterations, et sigma = 15:

```
pnonlocalerosion 15 10 8 tangram.pan out.pan
```

Voir aussi

pnonlocaldilation

Prototype C++

```
Errc PNonLocalErosion( const Img2duc & ims, Img2duc & imd, float
sigma, int nbIter, int ngbIdx );
```

Auteur: Matthieu Toutain

pnonlocalmeanfiltering

Lissage d’une image par un filtre moyenneur non linéaire.

Synopsis

pnonlocalmeanfiltering sigma nb_iter connectivity [*im_in*|-] [*im_out*|-]

Description

L’opérateur **pnonlocalmeanfiltering** permet d’effectuer un lissage moyenneur non lin"aire sur l’image *im_in*.

Soit f l’image *im_in*, l’algorithme applique itérativement:
$f(u)^{t+1} = \sum_{v ~ u}{w(u, v)^{p / 2} |f^t(v) - f^t(u)|^{p - 2} f(v)} / \sum_{v ~ u}{ w(u, v)^{p / 2} |f^t(v) - f^t(u)|^{p - 2}}$
avec u un pixel, v un voisin de u, $w(u, v)$ est le poids entre u et v (qui peut être une mesure de similarité, une distance, etc.).

Le paramètre *p* can be 1, 2, or any.

Le poids entre un pixel et ses voisins est calculé par:
$w(u, v) = \exp^{-||f(v) - f(u)||^2 / \sigma^2}$.

Paramètres

- *sigma*: seuil maximal de la mesure de similarité. Plus la valeur de sigma est élevée, moins fort est le filtage.
- *nb_iter*: le nombre d’itérations.
- *connectivity*: la connexité: 4 ou 8.

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: une image du même type que l’image d’entrée ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Applique un filtrage moyenneur non local à l'image tangram.pan, avec une 8-connectivité, 10 itérations, et un sigma = 15:

```
pnonlocalmeanfiltering 15 10 8 tangram.pan out.pan
```

Voir aussi

Filtrage

Prototype C++

```
template Errc lplRegularization(Imx2d &imgIn, Imx2d &imgOut, float
sigma, int connectivity, IRunThroughImage * imageRunner, IPerformReg
* regPerformer);
```

Auteur: Régis Clouard

pnonlocalmedianfiltering

Filtage médian non local d'une image.

Synopsis

pnonlocalmedianfiltering sigma nb_iter connectivity [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pnonlocalmedianfiltering** permet d'appliquer un filtrage non local de type médian. Soit f l'image *im_in*, l'algorithme applique itérativement:
$f(u)^{t+1} = Med_{v \sim u}{v \tilde u}(\sqrt{w(u, v)}(f^t(v) - f^t(u))) + f^t(u)$
avec u un pixel, v un voisin de u, $w(u, v)$ est le poids entre u et v (qui peut être une mesure de similarité, une distance, etc.).

Les poids sont calculés entre la valeur de pixel et ses voisins par: (ici avec un mesure de similarité exponentielle):
$w(u, v) = \exp^{-||f(v) - f(u)||^2 / \sigma^2}$
avec $w(u, v) = 1$, on a un filtre median classique.

Paramètres

- *sigma*: seuil maximal de la mesure de similarité. Plus la valeur de sigma est élevée, moins fort est le filtage.
- *nb_iter*: le nombre d'itérations.
- *connectivity*: la connexité: 4 ou 8.

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: Une image de meme type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Applique un filtrage median non local à l'image tangram.pan, avec une 8-connectivité, 10 itérations, et un sigma = 15:

  ```
  pnonlocalmedianfiltering 15 10 8 tangram.pan out.pan
  ```

Voir aussi

Filtrage

Prototype C++

```
template Errc lplRegularization(Imx2d &imgIn, Imx2d &imgOut, float
sigma, int connectivity, IRunThroughImage * imageRunner, IPerformReg
* regPerformer);
```

Author: Matthieu Toutain

pnonmaximasuppression

Suppression des points non maxima dans une image d'amplitude de gradient.

Synopsis

pnonmaximasuppression [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pnonmaximasuppression** permet de ne conserver de l'image d'amplitude du gradient *im_in1* que les valeurs qui sont maximales dans la direction orthogonale au gradient donné dans l'image *im_in2*.

L'image de sortie *im_out* sera de même type que l'image d'entrée et ne contient que les points maxima.

L'image *im_in2* contient pour chaque pixel une valeur dans l'intervalle [0..7] correspondant au codage de la direction en 8-voisinage selon le codage de Freeman. Les codes de Freeman sont :

```
  2D        3D
              z-1:          z:              z+1:
1 2 3       2  3  4      10 11 12         19 20 21
0   4       1  0  5      9     22         18 13 14
7 6 5       8  7  6      25 24 23         17 16 15
```

Une valeur de gradient n'est conservée dans l'image *im_out* que si ses deux voisins orthogonaux dans la direction du gradient lui sont inférieurs ou égaux.

De même, les valeurs du bord de l'image *im_in2* sont recopiées dans *im_out*.

Entrées

- *im_in1*: une image de niveaux de gris.
- *im_in2*: une image d'octets (Uchar image).

Sorties

- *im_out*: une image du même type que l'image *im_in1*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Détection de contours dans l'image tangram.pan:

```
pexponentialfiltering 0.7 tangram.pan i1.pan
pgradient 1 i1.pan i2.pan i3.pan
pnonmaximasuppression i2.pan i3.pan i4.pan
ppostthinning i4.pan i5.pan
pgradientthreshold 0.03 i2.pan
seuilhaut=‘pstatus‘
pbinarization $seuilhaut 1e30 i5.pan i6.pan
pgradientthreshold 0.2 i2.pan
seuilbas=‘pstatus‘
pbinarization $seuilbas 1e30 i5.pan  i7.pan
pgeodesicdilation 1 1 -1 i6.pan i7.pan out.pan
```

Voir aussi

Détection de contours

Prototype C++

```
Errc PNonMaximaSuppression( const Img2duc &im_in1, const Img2duc
&im_in2, Img2duc &im_out );
```

Auteur: Régis Clouard

pnormalization

Normalisation d'une image entre deux valeurs extrèmes.

Synopsis

pnormalization *min max* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pnormalization** calcule l'image *im_out* par normalisation des valeurs des pixels de l'image *im_in* entre les valeurs *min* et *max*.

La formule de calcul est la suivante :

```
pixel(im_out) = [(max - min) / (Max(im_in)-Min(im_in))] * pixel(im_in)
              + [(min*Max(im_in) - max*Min(im_in)) / (Max(im_in)-Min(im_in))];
```

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Le type de l'image de sortie est le même que celui de l'image d'entrée.

Paramètres

- *min* et *max* sont des valeurs du même type que l'image d'entrée (eg., 0..255 pour les images de Char).

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *im_out*: une image ou un graphe du même type que *im_in.*

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Normalisation des valeurs de pixels entre 10 et 234:

```
pnormalization 10 234 tangram.pan a.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PNormalization( const Img2duc &im_in, Img2duc &im_out, Uchar
min, Uchar max );
```

Auteur: Régis Clouard

pnot

Négation logique d'image ou de graphe et complémentaire d'une carte de régions.

Synopsis

pnot [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'oérateur **pnot** effectue la négation logique de l'image d'entrée.

Pour les images, la négation utilise l'opérateur C '!'. Si un pixel est supérieur à 0 alors sa valeur devient 0 sinon sa valeur devient 1.

```
pixel(im_out) = ! pixel(im_in)
```

Pour les images couleur ou multispectrale, la négation est appliquée sur chaque bande séparément.

Pour les graphes, la négation utilise l'opérateur C '!' et est appliquée sur les valeurs de noeud.

Pour les cartes de régions, **pnot** correspond au complémentaire des régions. Les régions avec un label > 0 deviennent la région de label=0 sinon la région 0 devient la région de label=1. En sortie les régions ne sont plus nécessairement connexes.

Entrées

- *im_in*: une image, un graphe ou une carte de régions.

Sorties

- *im_out*: un objet du même type que l'entrée.

Résultat

Retourne SUCCESS ou FAILURE.
Pour les cartes de régions, retourne la valeur de label maximale (0 or 1).

Exemples

- Sélectionne le fond de l'image tangram :

```
pbinarization 100 1e30 examples/tangram.pan a.pan
plabeling 8 a.pan b.pan
pnot b.pan c.pan
```

Voir aussi

Logique

Prototype C++

```
Errc PNot( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

pobject2col

Création d'une collection contenant un objet Pandore.

Synopsis

pobject2col *name* [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pobject2col** crée une collection contenant l'objet Pandore *im_in* sous le nom *name*.

Paramètres

- *name* est le nom de l'objet Pandore dans la collection.

Entrées

- *im_in*: un objet Pandore.

Sorties

- *im_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Ajoute l'image tangram.pan dans la collection col.pan :

```
pobject2col foo tangram.pan col.pan
pfile col.pan
```

Voir aussi

Collection

Prototype C++

```
Errc PObject2Col( const Img2duc &obj_in, Collection &col_out, const
std::string &name );
```

Auteur: Alexandre Duret-Lutz

popencontourselection

Sélection de chaînes de contours ouvertes sur leur longueur.

Synopsis

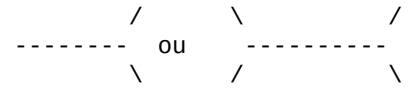
popencontourselection *relation longueur* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **popencontourselection** consiste à supprimer toutes les chaînes de contours **ouvertes** sur leur valeur de longueur. C'est le paramètre *relation* qui indique le type de la relation d'ordre.

Une chaîne ouverte est une séquence continue en 8 connexité en 2D (ou 26 connexité en 3D) de pixels non nuls, d'épaisseur 1, qui commence sur un pixel terminal ou sur une intersection et qui s'arrête sur un pixel terminal ou sur une intersection avec une autre chaîne.

Une chaîne fermée est considérée ici comme une chaîne, de même que les lignes et les barbules sont des chaînes:



Attention: les points terminaux sont des points qui ne possèdent qu'un voisin. Il peut alors être utile de faire précéder cet opérateur d'opérateurs d'amincissement qui garantissent la 8-connexité (ex: **ppostthinning**).

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de longueur.
  - *relation* = 3: contours avec la longueur maximale.
  - *relation* = 2: contours >= *longueur*.
  - *relation* = 1: contours > *longueur*.
  - *relation* = 0: contours = *longueur*.
  - *relation* = -1: contours < *longueur*.
  - *relation* = -2: contours <= *longueur*.
  - *relation* = -3: contours avec la longueur minimale.
- La longueur *longueur* est comptée en nombre de pixels.

Entrées

- *im_in*: une image de type Uchar.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Le nombre de chaînes supprimées.

Exemples

Sélectionne les contours ouverts obtenus à partir d'une simple détection de contours :

```
psobel tangram.pan b.pan
pbinarization 45 1e30 b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
popencontourselection 1 5 e.pan out.pan
pstatus
```

Voir aussi

Contour

Prototype C++

```
Errc POpenContourSelection( const Img2duc &im_in, Img2duc &im_out,
int relation, int longueur );
```

Auteur: Régis Clouard

por

Ou binaire entre images ou graphes, et union entre cartes de régions.

Synopsis

por [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **por** effectue le ou bit à bit entre les deux images d'entrée *im_in1* et *im_in2*.

Pour les images de réelles, le "ou" s'implante avec l'opérateur C '|' et s'applique sur chaque pixel :

```
pixel(im_out) = pixel(im_in1) | pixel(im_in2);
```

Pour les images réelles, le "ou" est:

```
pixel(im_out) = pixel(im_in1) + pixel(im_in2);
```

Pour les images couleur et multispectrale, le "ou" est calculé sur chacune des bandes séparément.

Pour les graphes, l'opérateur "ou" s'implante par l'opérateur C + entre les valeurs de noeud.

Pour les cartes de régions, le "ou" correspond à l'union des régions. La carte de régions résultante *im_out* est composée des régions de chacune des deux cartes de régions d'entrée en donnant préférence aux plus petites régions en cas de superposition.

Les deux entrées doivent être de même type.

Entrées

- *im_in1*: une image, un graphe ou une carte de régions.
- *im_in2*: une image, un graphe ou une carte de régions.

Sorties

- *im_out*: un objet du même type que im_in1 et im_in2.

Résultat

Retourne SUCCESS ou FAILURE.

Pour les cartes de région, retourne la valeur de label maximum.

Exemples

- Superposition des fronti&egraveres des pièces de tangram sur l'image originale :

```
pbinarization 100 1e30 examples/tangram.pan a.pan
pboundary 8 a.pan b.pan
por examples/tangram.pan b.pan c.pan
```

Voir aussi

Logique

Prototype C++

```
Errc POr( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

porientationselection

Sélection de régions sur leur valeur d'orientation.

Synopsis

porientationselection *relation seuil* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **orientationselection** permet de sélectionner les régions sur leur orientation. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

La valeur d'orientation est donnée en degré [0,360].
Elle est calculée à partir des moments d'inertie:

```
orientation = 0.5*arctan(2*M11/(M20-M02)).
```

Si `M20=M02` alors c'est que la région présente une symétrie de rotation. Dans ce cas, la valeur d'orientation est égale à **360**.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur en degré [0..360] par rapport à l'axe d'inertie.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de régions ainsi sélectionnée.

Exemples

Sélectionne les régions verticales :

```
porientationselection 90 a.pan b.pan
```

Voir aussi

Région

Prototype C++

```
Errc PAreaOrientation( const Reg2d &rg_in, Reg2d &rg_out, int
relation, Ushort seuil );
```

Auteur: Régis Clouard

poutborderselection

Sélection des régions qui ne touchent pas le bord de l'image.

Synopsis

poutborderselection *l h p* [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **poutborderselection** construit une nouvelle carte de régions *rg_out* avec les régions de *rg_in* qui n'ont pas de pixel touchant le bord de l'image.

Le bord est caractérisé par une profondeur de *p*, une hauteur de *h* et une longueur de *l* pixels.

Les régions ne sont pas reétiquetée. Elles gardent la même valeur de label que dans la carte d'entrée.

Paramètres

- *l, h p* définissent la taille du bord.

Entrées

- *rg_in*: une carte de régions

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de régions restantes.

Exemples

Supprime les régions qui touchent le bord de la carte de région :

```
poutborderselection 1 1 0 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc POutBorderSelection( const Reg2d &rg_in, Reg2d &rg_out, int l,
int h, int d );
```

Auteur: Régis Clouard

poutrangefiltering

Lissage par filtre adaptatif basé sur le choix des voisins.

Synopsis

poutrangefiltering *difference* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **poutrangefiltering** permet d'appliquer un lissage linéaire en respectant les points de fort contraste.

Le principe est de remplacer chaque point par la valeur moyenne de ses voisins, si l'écart entre cette moyenne et le point central est inférieur au seuil *difference*.

Paramètres

- *difference* est une valeur réelle qui représente l'ecart maximum toléré de la moyenne des voisins è la valeur du point central. Si *difference*=0 alors **poutrangefiltering** correspond à un lissage moyenneur.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applique un lissage de type out range à l'image tangram.pan :

```
poutrangefiltering 10 tangram.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc POutRangeFiltering( const Img2duc &im_in, Img2duc &im_out,
float difference );
```

Auteur: Régis Clouard

ppan2analyze

Conversion d’une image Pandore en image ANALYZE 7.5.

Synopsis

ppan2analyze *im_in* [*im_out*|-]

Description

L’opérateur **ppan2analyze** produit une image ANALYZE 7.5 à partir d’une image Pandore.

Une image ANALYZE 7.5 est composée de 2 fichiers avec la même nom de base et dans le même dossier:

- un fichier d’entête (suffixé .hdr)
- un fichier d’image (suffixé .img).

im_out est le nom de base de l’image ANALYZE sans suffixe.

Entrées

- *im_in*: une image Pandore.

outputs

- *im_out*: le nom de base de l’image ANALYZE 7.5.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l’image a.pan en image ANALYZE "brain.hdr" et "brain.img".

```
pan2analyze a.pan brain
```

Voir aussi

Conversion

Prototype C++

```
Errc Pan2Analyze( const Imx3d &img, const char *filename_out );
```

Auteur: David Tschumperlé

ppan2bmp

Conversion d'une image 2D Pandore en image BMP.

Synopsis

ppan2bmp [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ppan2bmp** permet de convertir une image de type *Pandore* en un fichier *bmp*.

Les seules images transformables sont:

- les images de niveaux de gris 2D d'octets;
- les images couleurs 2D d'octets;
- les cartes de régions 2D.

Les autres types d'images nécessitent des conversions adaptées au préalable.

Entrées

- *im_in*: une image 2D (img2duc, Imc2duc) ou une carte de régions 2D.

Sorties

- *im_out*: un fichier BMP.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
ppan2bmp tangram.pan image.bmp
```

Voir aussi

Conversion, pbmp2pan

Prototype C++

```
Errc PPan2Bmp( const Img2duc &im_in, char *filename );
```

Auteur: Régis Clouard

ppan2d23d

Conversion d'une série d'images 2D en une image 3D.

Synopsis

ppan2d23d *first-image-number last-image-number im_in_template* [*im_out*|-]

Description

L'opérateur **ppan2d23d** transforme une série d'images 2D allant de *first-image-number* jusqu'a *last-image-number* en une image 3D unique. Chaque image 2D constitue un plan de l'image 3D.

Le nom de l'image d'entrée *im_in_template* est utilisé pour spécifier le nom des fichiers réels à prendre en compte. Les # dans le nom de l'*im_in_template* sont utilisés pour spécifier le format des numéros des images 2D. Par exemple:

- `toto####.pan` pour les bornes 8 et 10, précisent que les fichiers 2D se nomment: `toto0008.pan, toto0009.pan, toto0010.pan`.
- `toto#.pan` pour les bornes 8 et 10, précisent que les fichiers 2D se nomment: `toto8.pan, toto9.pan, toto10.pan`.

Paramètres

- *first-image-number* et *last-image-number* représentent le numéro du premier et dernier fichier correspondant au premier et dernier plans de l'image 3D. Il n'y a pas de relation directe entre les nombres et les numéros de plans. La première image 2D devient le plan 0, et l dernière image le dernier plan.

- *im_in_template* est le préfixe du noms des images d'entrée. Il utilise le caractère # pour spécifier le format du nombre dans le nom.

Entrées

- *im_in_template*: une image 2D.

Sorties

- *im_out*: une image Pandore 3D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
ppan2d23d 0 10 image##.pan image3d.pan
```

Voir aussi

Conversion

Prototype C++

```
Errc PPan2d23d( const char *nom2d, const char *nom3d, int
first-image-number, int last-image-number );
```

Auteur: François Angot

ppan2fits

Conversion d'une image Pandore vers le format FITS.

Synopsis

ppan2fits [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ppan2fits** construit une image FITS (Flexible Image Transport System) à partir d'une image Pandore.

Seules les images de niveaux de gris sont prises en compte dans la version actuelle.

Entrées

- *im_in*: une image Pandore.

Sorties

- *im_out*: une image FITS.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image Pandore en image FITS :

```
ppan2fits tangram.pan tangram.fits
```

Voir aussi

Conversion, pfits2pan

Prototype C++

```
Errc PPan2Fits( const Img3dsf &ims, char *filename );
```

Auteur: Jalal Fadili

ppan2gif

Conversion d’une image pandore 2D en image GIF.

Synopsis

ppan2gif [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L’opérateur **ppan2gif** permet de convertir une image de type *Pandore* en un fichier *GIF*.

Seuls les types d’image Pandore suivants sont pris en compte:

- image de niveaux de gris 2D d’octets (Img2uc);
- image couleur 2D d’octets (Imc2duc);
- carte de régions 2D (Reg2d).

Les autres types d’image Pandore doivent être convertis en utilisant les opérateurs de coercition appropriés.

Entrées

- *im_in*: une image 2D (img2duc, Imc2duc) ou une carte de régions 2D.

Sorties

- *im_out*: un fichier GIF.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
ppan2gif tangram.pan image.gif
```

Voir aussi

Conversion, pgif2pan

Prototype C++

```
Errc PPan2Gif( const Img2duc &im_in, char *filename );
```

Auteur: Régis Clouard

ppan2jpeg

Conversion d'une image 2D Pandore en image JPEG.

Synopsis

ppan2jpeg quality [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ppan2jpeg** permet de convertir une image de type *Pandore* en un fichier *jpeg*.

Les seules images transformables sont:

- les images de niveaux de gris 2D d'octets;
- les images couleurs 2D d'octets;

Les autres types d'images nécessitent des conversions adaptées au préalable.

Paramètre

- *quality*: la qualité est exprimée par un nombre rèel sur une échelle de pourcentage [0..1], où 1 correspond à la meilleure qualité (sans dégradation).

Entrées

- *im_in*: une image 2D (img2duc, Imc2duc) ou une carte de régions 2D.

Sorties

- *im_out*: un fichier JPEG.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
ppan2jpeg 0.9 tangram.pan image.jpeg
```

Voir aussi

Conversion, pjpeg2pan

Prototype C++

```
Errc PPan2Jpeg( const Imc2duc &ims, char *fich, float quality );
```

Auteur: Régis Clouard

ppan2pan

Conversion d’un fichier Pandore en un fichier Pandore.

Synopsis

ppan2pan [*im_in*|-] [*im_out*|-]

Description

L’opérateur **ppan2pan** permet de convertir un fichier Pandore d’une ancienne version. Les anciens format Pandore sont les formats Pandore2 et Pandore3.

Les anciennes versions ne sont plus lisibles par la nouvelle version pour des raisons de rapidité de chargement et d’optimisation du code de la "library".

Entrées

- *im_in*: un fichier Pandore.

Sorties

- *im_out*: un objet de même type que l’objet d’entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit une image d’un ancien format Pandore en image de format courant :

```
ppan2pan oldtangram.pan tangram.pan
```

Voir aussi

Conversion

Prototype C++

```
Errc PPan2pan( const FILE* fdin, Pobject **objout );
```

Auteur: Régis Clouard

ppan2png

Conversion d'une image 2D Pandore en image PNG.

Synopsis

ppan2png [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ppan2png** permet de convertir une image de type *Pandore* en un fichier *PNG*.

Les seules images transformables sont:

- les images de niveaux de gris 2D d'octets;
- les images couleurs 2D d'octets;

Les autres types d'images nécessitent des conversions adaptées au préalable.

Entrées

- *im_in*: une image 2D (img2duc, Imc2duc) ou une carte de régions 2D.

Sorties

- *im_out*: un fichier PNG.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
ppan2png tangram.pan image.png
```

Voir aussi

Conversion, ppng2pan

Prototype C++

```
Errc PPan2Png( const Imc2duc &ims, char *fich );
```

Auteur: Régis Clouard

ppan2ppm

Conversion d'un fichier Pandore en un fichier de format PPM (Portable PixMap ASCII) ou PGM (Portable GrayMap ASCII).

Synopsis

ppan2ppm [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ppan2ppm** permet de transformer une image Pandore en une image au format PPM ASCII.

Un fichier PPM consiste en 2 parties, un entête et les données:

- L'entête consiste en 3 parties délimitées par un retour chariot ou par une fin de ligne (bien que la norme n'exige que des espaces):
  - La première ligne contient le magic number.
    - P2: pour une image en niveaux de gris avec les données en ASCII.
    - P5: pour une image en niveaux de gris avec les données en binaire.
    - P3: pour une image couleur avec les données en ASCII.
    - P6: pour une image couleur avec les données en binaire.
  - La seconde ligne donne le nombre de colonnes puis le nombre de lignes en ASCII.
  - La dernière partie donne le nombre de couleurs maximales utilisées.

  Chaque ligne peut contenir un commentaire introduit par #.

- Le format des données dépend du magic number.
  - P2: les pixels sont codés en ASCII sous la forme d'un nombre entier (le niveau de gris).
  - P5: les pixels sont codés en binaire sous la forme d'un nombre entier (le niveau de gris).
  - P3: les pixels sont codés en ASCII sous la forme de trois nombres consécutifs (Rouge Vert Bleu).
  - P6: les pixels sont codés en binaire sous la forme de 3 octets consécutifs (Rouge Vert Bleu).

Exemple:

```
P2
#Image en niveaux de gris
#de 5 colones et de 4 lignes.
5 4
255
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

Astuce: Le format PPM permet de construire facilement un fichier texte utilisable par d'autres logiciels tel que gnuplot

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: un fichier PPM (couleur) ou PGM (niveaux de gris).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Convertit l'image tangram en image ppm.

```
ppan2ppm tangram.pan tangram.ppm
```

Voir aussi

Conversion, pppm2pan

Prototype C++

```
Errc PPan2PPM( const Img2duc &ims, char* f_out );
```

Auteur: Régis Clouard

ppan2ps

Conversion d’une image Pandore en un fichier Encapsuled PostScript.

Synopsis

ppan2ps [*im_in*|-] [*file_out*|-]

Description

L’opérateur **ppan2ps** permet de convertir une image Pandore en un fichier PostScript.

Seules les images 2D et les cartes de règions 2D peuvent être converties en fichier PostScript.

Le fichier résultant peut alors être imprimé ou intégré dans un document Word, TeX, Excel...

Entrées

- *im_in*: une image 2D ou une carte de régions 2D.

Sorties

- *file_out*: un fichier PostScript.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit image PostScript à partir de l’image Pandore tangramp.pan :

```
ppan2ps tangram.pan tangram.eps
```

Voir aussi

Conversion

Prototype C++

```
Errc PPan2PS( const Img2duc &im_in, char *filename );
```

Auteur: Régis Clouard

ppan2raw

Conversion d’un fichier image Pandore en un fichier image sans entête.

Synopsis

ppan2raw [*im_in*|-] [*file_out*|-]

Description

L’opérateur **ppan2raw** permet de transformer une image au format Pandore en une image sans entête (Format RAW). Les fichiers sont des fichiers avec des donées binaires :

- Les pixels sont écrits les uns derrière les autres selon leur taille.
- Pour les images 2D, les données sont fournies par ordre des lignes: d’abord tous les pixels de la ligne 0, puis de la ligne 1...
- Pour les images 3D, les données sont fournies par ordre des plans, d’abord tous les pixels du plan 0, puis du plan1 ...
- Pour les images couleur, les données sont fournies par ordre des bandes, d’abord tous les pixels de la bande Rouge, puis Verte puis Bleue.

Entrées

- *im_in*: une image Pandore.

Sorties

- *file_out*: un fichier binaire.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
ppan2raw tangram.pan tangram.raw
```

Voir aussi

Conversion, praw2pan

Prototype C++

```
Errc Raw2Pan( const Img2duc &im_in, char *filename );
```

Auteur: Régis Clouard

ppan2tiff

Conversion d'une image Pandore en image(s) TIFF.

Synopsis

ppan2tiff [*-m mask*] [*im_in*|-] *im_out*|-]

Description

L'opérateur **ppan2tiff** convertit une image Pandore en image TIFF. La conversion **ppan2tiff** peut générer des fichiers tiff en 8 et 16 bits correspondant respectivement aux images char et Long de Pandore, que ce soit en niveaux de gris ou en couleur.

Dans le cas d'une image Long (Img2dslm, Img3dsl, Imc2dsl ou Imc3dsl), les pixels de l'image source *im_in* codés sur 4 octets sont écrétés sur 2 octets (0..65535). Tout ce qui est supérieur à 65535 est ramené à 65535. Tout ce qui est inférieur à 0 est ramené à 0. Il peut alors être utile d'utiliser les opérateurs de recadrage des valeurs avant d'utiliser cet opérateur.

Dans le cas d'une image 3D, **ppan2tiff** construit autant de fichiers TIFF que de plans dans l'image Pandore. Le nom des fichiers de sorties seront de la forme: "`im_out`*ndep*.tif".
Par exemple, la commande **ppan2tiff** a.pan img.tiff, produira des fichiers du genre `img000.tif, img001.tif ...`

Entrées

- *im_in*: une image ou une carte de régions.

Sorties

- *im_out*: un fichier TIFF.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
ppan2tiff tangram.pan image.tiff
```

Voir aussi

Conversion, ptiff2pan

Prototype C++

```
Errc PPan2Tiff( const Img2duc &im_in, char *im_out );
```

Auteur: Régis Clouard

ppan2txt

Conversion d'une image en une liste de points dans un fichier texte.

Synopsis

ppan2txt [*-m mask*] [*im_in*|-] [*file_out*|-]

Description

Cet opérateur permet de construire un fichier texte contenant la liste des **points non nuls** de l'image *im_in*. Le fichier texte *fichier_texte* aura la structure suivante (pour une image 3d):

```
valeur x y z
```

ou comme suit pour une image 2D :

```
valeur x y
```

Entrées

- *im_in*: une image de niveaux de gris (1D, 2D ou 3D).

Sorties

- *file_out*: un fichier texte.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
ppan2txt tangram.pan image.txt
```

Voir aussi

Conversion, ptxt2pan

Prototype C++

```
Errc Pan2txt( const Img3duc &im_out, char *nom, Long z, Long y Long
x );
```

Auteur: Régis Clouard

ppan2vff

Conversion d'une image Pandore 3D en une image au format VFF.

Synopsis

ppan2vff [*-m mask*] [*im_in*|-] [*file_out*|-]

Description

L'opérateur **ppan2vff** convertit une image Pandore en une image VFF (sunvision).

Une image au format VFF est construite à partir de l'image 3D d'entrée.

Pour une image d'octets non signés, la conversion est une recopie. Pour d'autres types d'image Pandore 3D, un recadage linéaire entre 0..255 est effectuée. Il peut donc avoir une très forte perte d'information.

Entrées

- *im_in*: une image 3D ou une carte de régions 3D.

Sorties

- *file_out*: un fichier VFF.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convrtit l'image Pandore cube.pan en image vff:

```
ppan2vff cube.pan image.vff
```

Voir aussi

Conversion, pvff2pan

Prototype C++

```
Errc PPan2Vff( const Img3duc &im_in, char *filename );
```

Auteur: François Angot

ppan3d22d

Conversion d'une image Pandore 3D en une série d'images Pandore 2D.

Synopsis

ppan3d22d [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ppan3d22d** transforme un image 3D en une série d'images Pandore 2D.

Le nom des fichiers de sortie est composé du
nom de base+le numéro du plan+le suffixe s'il existe.

Par exemple, la commande `pan3d22d a.pan b.pan` produit les images:

- si `a.pan` contient 13 plans : `b00.pan, b01.pan ... b12.pan`
- if `a.pan` contient 121 plans : `b000.pan, b001.pan ... b120.pan`

Entrées

- *im_in*: une image Pandore.

Sorties

- *im_out*: une image Pandore 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
ppan3d22d cube.pan image.pan
```

Voir aussi

Conversion, ppan2d23d

Prototype C++

```
Errc PPan3d22d( const Img3duc &im_in, char *filename );
```

Auteur: François Angot

pparrec2pan

Conversion d’une image au format PAR/REC (Philips Medical System) en image Pandore.

Synopsis

pparrec2pan *im_in* [*im_out*|-]

Description

L’opérateur **pparrec2pan** convertit une image PAR/REC en image Pandore.

Une image PAR/REC est composée de 2 fichiers:

- un fichier entête (suffixed .par)
- un fichier de données (suffixed .rec).

Le fichier d’entrée *im_in* est un des deux fichiers PAR/REC files. Le second fichier est lu dans le même répertoire en utilisant le même nom de base avec le suffixe approprié.

L’image de sortie *im_out* est une image 3D de float (Imx3dsf).

Entrées

- *im_in*: un fichier PAR/REC (soit .par soit .rec file)

Sorties

- *im_out*: une image Pandore (Imx3dsf).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Conversion de l’image "brain" en image Pandore "a.pan" puis affichage de la bande 0.

```
parrec2pan brain.par a.pan
pimx2img 0 a.pan | pvisu
```

Voir aussi

Conversion

Prototype C++

```
Errc PParrec2Pan( const char *filename, Pobject **obj_out );
```

Avertissement

Auteur: David Tschumperlé

ppeergroupfiltering

Lissage d'une image couleur par Peer Group Filtering.

Synopsis

ppeergroupfiltering *amplitude* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ppeergroupfiltering** permet de lisser une image couleur par Peer Group Filtering. Un pixel est remplacé par la somme pondérée (par une gaussienne) de ses pixels voisins proches en terme de distance couleur.

Paramètres

- *amplitude* donne l'amplitude du bruit à ne pas considérer.

Entrées

- *im_in*: une image couleur.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applies the Peer Group Filtering filter on parrot.pan image:

```
ppeergroupfiltering 10 parrot.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PPeerGroupFiltering( const Imc2duc &im_in, Imc2duc &im_out,
float amplitude );
```

Auteur: Olivier Lezoray

pperimeterselection

Sélection de régions sur leur valeur de périmètre.

Synopsis

pperimeterselection *relation seuil* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pperimeterselection** permet de sélectionner les régions sur leur valeur de périmètre. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

Le périmètre correspond au nombre de points internes d'une région qui sont sur la frontière de la région. Il est calculée. nombre de pixels de la frontière. A chaque creux interne dans une région, on ajoute un petit quart de pixel de périmètre. Dans l'exemple suivant, le périmètre est égal à 7= 6+ 4x.25.

```
 xx
xxxx
 xx
```

Pour une région de 1 pixel, le périmètre est égal à 1.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur entière exprimée en nombre de pixels.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de régions ainsi sélectionnée.

Exemples

Sélectionne les régions avec une frontière > 50 pixels:

```
pperimeterselection 1 50 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PPerimeterSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, Ulong seuil );
```

Auteur: Régis Clouard

pphase

Calcul de la phase entre deux images.

Synopsis

pphase [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pphase** calcule la phase des valeurs de niveaux de gris entre chaque pixel de l'image *im_in1* et de l'image *im_in2* correspondant. Le résultat est mis dans l'image destination *im_out* qui est de type réel.

Cet opérateur est notamment utilisé pour calculer le phase d'une image complexe (partie réelle et partie imaginaire).

La formule de calcul est la suivante :

```
pixel(im_out) = atan(pixel(im_in2)/*pixel(im_in1))
```

Entrées

- *im_in1*: une image de niveaux de gris ou de couleur.
- *im_in2*: une image de niveaux de gris ou de couleur.

Sorties

- *im_out*: une image de float.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit une image de carré du domaine spatial dans le domaine fréquentiel et réciproquement :

```
pshapedesign 256 256 0 2 20 0 square.pan
pshapedesign 256 256 0 0  0 0 empty.pan
pfft square.pan empty.pan real.pan imaginary.pan
pmodulus real.pan imaginary.pan modulus.pan
pphase real.pan imaginary.pan phase.pan
pifft real.pan imaginary.pan square1.pan empty1.pan
plineartransform 0 0 255 square1.pan square2.pan
pim2uc square2.pan newsquare.pan
```

Voir aussi

Domaine Fréquentiel, pmodulus

Prototype C++

```
Errc PPhase( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

ppixelvalue

Affiche la valeur d'un pixel d'un image ou d'un noeud d'un graphe donné.

Synopsis

ppixelvalue *x y z* [*-m mask*] [*im_in*|-] [*col_out*|-]

Description

L'opérateur **ppixelvalue** permet d'obtenir la valeur du pixel de coordonnées (x,y,z) en 2D ou du voxel de coordonnées (x,y,z) en 3D dans l'image d'entrée *im_in*.

Pour un graphe, l'opérateur permet d'afficher la valeur du x_{eme} noeud.

La valeur est ensuite récupérable (et affichable) par l'opérateur **pstatus**.

Les valeurs de pixel de chaque bande sont stockées dans la collection *col_out*.

Paramètres

- *x*, *y*, *z* correspondent è la coordonnée du pixel ou voxel dans l'image. Pour les images 2D, *z* est ignoré mais doit être donné.

Entrées

- *im_in*: une image, une carte de régions ou un graphe.

Sorties

- *col_out*: une collection de valeurs réelles.

Résultat

Retourne la valeur de pixel ou FAILURE si les coordonnées n'appartiennent pas à l'image d'entrée (pour la première bande uniquement). Cette valeur est accessible par la commande **pstatus**.

Exemples

Retourne la valeur au coordonnées 20,50 dans tangram.pan (version Unix):

```
ppixelvalue 20 50 0 tangram.pan col.pan
val=‘pstatus‘
echo "valeur = $val"
```

Retourne la valeur au coordonnées 20,50 dans tangram.pan (version MsDOS):

```
ppixelvalue 20 50 0 tangram.pan col.pan
call pstatus
call pset val
echo valeur = %val%
```

Voir aussi

Utilitaire

Prototype C++

```
Float PPixelValue( const Img3duc &im_in, Collection & col_out, Long z, Long y, Long x );
```

Auteur: Régis Clouard

pplot1d

Construction d'une image 2D à partir d'une fonction 1D.

Synopsis

pplot1d *dimx dimy type ymin ymax* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pplot1d** dessine une représentation 2D de fonction 1D. Les paramètres *dimx* et *dimy* spécifient les dimensions de l'image de sortie.

La représentation peut prendre plusieurs formes en fonction du paramètre *type*.

Paramètres

- *dimx* spécifie la largeur de la représentation 2D.
- *dimy* spécifie la hauteur de la représentation 2D.
- *type* spécifie le type de représentation parmi:
  - 0 = lignes (continue par morceaux).
  - 1 = barres de diagramme.
  - 2 = lignes (interpolation bicubique).
- *ymin* spécifie la valeur minimale sur l'axe y.
- *ymax* spécifie la valeur maximale sur l'axe y. **Note**; Si ymin=ymax=0 alors l'échelle est automatiquement calculée à partir des valeurs d'entrée.

Entrées

- *im_in*: une image 1D de niveaux de gris.

Sorties

- *im_out*: une image 2D couleur (Imc2duc)

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche l’histogramme de l’image tangram.pan:

```
phistogram tangram.pan a.pan
pplot1d 512 256 0 0 0 a.pan b.pan
pvisu b.pan
```

Voir aussi

Visualisation

Prototype C++

```
Errc PPlot1d( const Img1d &ims, Imc2duc &imd, Uchar type,Float ymin,
Float ymax );
```

Avertissement

Auteur: D. Tschumperlé

pplotquiver

Dessine un champ de vecteurs 2D à partir d'une image 2D multispectrale à deux composantes.

Synopsis

pplotquiver *dimx dimy sampling factor* [*im_in*|-] [*im_out*|-]

Description

Cet opérateur permet de dessiner un champ de vecteurs 2D à partir d'une image 2D multispectrale à deux bandes.

La premiere bande de l'image multispectrale contient la composante x du vecteur et la seconde la composante y. Cet image multispectrale peut être créée avec l'opérateur: pregistrationPDE

Paramètres

Les paramètres de l'opérateur contrôlent le type de rendu :

- *dimx* : Définit la largeur de l'image du rendu.
- *dimy* : Définit la hauteur de l'image du rendu.
- *sampling* : Définit l'intervalle (en pixels) entre deux tracés de vecteurs consécutifs.
- *factor* : Définit un facteur multiplicatif appliqué à la longueur des vecteurs. Si la valeur est négative, elle correspond à un pourcentage de la longueur maximum rencontrée dans l'image de vecteurs d'entrée.

Entrées

- *im_in*: une image multispectrale 2D avec au moins 2 bandes.

Sorties

- *im_out*: une image 2D de Uchar (Img2duc).

Résultat

Retourne SUCCESS ou FAILURE.

Exemple

- Produit une translation horizontale de l'image "tangram.pan" puis affiche les vecteurs de déplacements

```
translation 0 10 tangram.pan tangram1.pan
pregistrationPDE 0.1 0.9 tangram.pan tangram1.pan a.pan
pplotquiver 800 800 10 -20 a.pan| visu
```

Voir aussi

Mouvement

Prototype C++

```
Errc PPlotQuiver(Imx2d &ims,Img2duc &imd,Short sampling,Float
factor);
```

Avertissement

Auteur: D. Tschumperlé

ppng2pan

Conversion d'une image PNG en image Pandore.

Synopsis

ppng2pan *im_in* [*im_out*|-]

Description

L'opérateur **ppng2pan** permet de convertir une image de type *PNG* en un fichier *Pandore*. Le fichier résultant est de type:

- Img2duc pour les images en niveaux de gris,
- Imc2duc pour les images couleur.

Entrées

- *im_in*: un fichier PNG.

Sorties

- *im_out*: une image Pandore.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
ppng2pan image.png image.pan
```

Voir aussi

Conversion, ppan2png

Prototype C++

```
Errc PPng2Pan( const FILE* fdin, Pobject **objout );
```

Auteur: Régis Clouard

ppolygonalapproximation

Approximation polygonale des contours d'une image.

Synopsis

ppolygonalapproximation *ecart* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ppolygonalapproximation** consiste à approximer les chaînes de contours par des polygones. Une chaîne est une séquence continue de pixels > 0 reposant sur un fond =0. De plus, ces chaînes doivent être d'épaisseur =1 pour donner un résultat optimal.

L'image de sortie *im_out* est une image de droites, chacune d'elles étiquetée avec une couleur différente.

Paramètres

- L'*écart* permet de préciser la distance aux chaînes de contours d'origine maximale autorisée. Il définit ainsi la précision de l'approximation.

Entrées

- *im_in*: une image de type Uchar.

Sorties

- *im_out*: une image de Uchar.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Effectue une approximation polygonale des frontières de pièces de tangram :

```
pbinarization 100 1e30 tangram.pan a.pan
pboundary 8 a.pan b.pan
ppostthinning b.pan c.pan
ppolygonalapproximation 5 c.pan d.pan
pbinarization 1 1e30 d.pan out.pan
```

Voir aussi

Contour

Prototype C++

```
Errc PPolygonalApproximation( const Img2duc &im_in, Img2duc &im_out,
Short ecart );
```

Auteur: Serge Coudé

ppolynomialfitting

Calcul de l'approximation du fond d'une image en utilisant une approximation polynomiale.

Synopsis

ppolynomialfitting *xorder yorder xyorder* [*im_in*|-] [*im_mk*|-] [*im_out*|-]

Description

ppolynomialfitting convertit le contenu d'une image en un fond homogène en utilisant une approximation polynomiale.

L'image *im_mk* est utilisée comme masque, et définit la liste des seuls pixels qui peuvent être utilisés pour calculer l'approximation polynomiale.

Les ordres du polynôme peuvent être sélectionnés séparemment pour x, y et xy. Par exemple, avec les ordres 2, 3, et 2 pour x, y, et xy respectivement, le polynôme sera:

```
a+b*x+c*x^2+d*y+e*y^2+f*y^3+g*xy
```

Paramètres

- *xorder* donnee degré pour x [0..10].
- *yorder* donne le degré pour y [0..10].
- *xyorder* donne le dehgré pour xy [0..10].

Entrées

- *im_in*: une image 2D.
- *im_mk*: une image d'octets utilisée comme masque.

Sorties

- *im_out*: une image 2D de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Correction de l'illumination de l'image tangram en utilisant la soustraction de fond. Le fond d'images est approximé par une expression polynomiale d'ordre 2:

```
pthresholding 0 73 tangram.pan mask.pan
ppolynomialfitting 2 2 1 tangram.pan mask.pan a.pan
pim2sf tangram.pan tangramf.pan
psub tangramf.pan a.pan b.pan
pmeanvalue a.pan; mean=‘pstatus‘
paddcst $mean b.pan out.pan
```

Autres exemples

Voir aussi

Surface Fitting

Prototype C++

```
Errc PPolynomialFitting( const Imx2d &im_in, const Img2d &im_mk,
Imx2d &im_out, int xOrder, int yOrder, int xyOrder );
```

Auteur: Régis Clouard

ppostthinning

Suppression des points de contours qui ne garantissent pas la 8 connexité (ou 26 connexité).

Synopsis

ppostthinning [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ppostthinning** consiste à supprimer tous les points de contours qui ne sont pas utiles à la préservation de la 8-connexité (ou de la 26-connexité en 3D).
Les points de contours superflus nuisent à la plupart des algorithmes de traitement de contours. C'est pourquoi cette commande précède généralement l'utilisation des opérateurs sur contours.

Un contour est une chaîne de pixels non nuls en 8-connexité (ou 26-connexité) reposant sur un fond nul.

Un point de contour "x" est éliminé (remplacé par "0") s'il ne détruit pas la 8-connexité. Par exemple, le centre est éliminé dans les cas suivants:

```
  |x|       | |0
 x|x|   ou x|x|   ou toutes autres symétries.
  | |0      |x|
```

Attention: Cet opérateur fonctionne sur des contours d'épaisseur 1 pixel. Il peut être nécessaire de faire précéder cet opérateur d'une squelettisation des contours (ex: pskeletonization).

L'image de sortie est du même type que l'image d'entrée.

Entrées

- *im_in*: une image 2D de type Uchar.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Extrait les contours de l'image tangram.pan :

```
pexponentialfiltering 0.7 tangram.pan i1.pan
pgradient 1 i1.pan i2.pan i3.pan
pnonmaximasuppression i2.pan i3.pan i4.pan
ppostthinning i4.pan i5.pan
pgradientthreshold 0.03 i2.pan
seuilhaut=‘pstatus‘
pbinarization $seuilhaut 1e30 i5.pan i6.pan
pgradientthreshold 0.2 i2.pan
seuilbas=‘pstatus‘
pbinarization $seuilbas 1e30 i5.pan  i7.pan
pgeodesicdilation 1 1 -1 i6.pan i7.pan out.pan
```

Voir aussi

Contour

Prototype C++

```
Errc PPostThinning( const Img2duc &im_in, Img2duc &im_out ); >
```

Auteur: Régis Clouard

ppow

Puissance nième d'une image ou d'un graphe.

Synopsis

ppow *n* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ppow** construit la puissance *n*ième. d'une image. Chaque pixel de l'image de sortie *im_out* est construit avec la puissance nième du pixel correspondant dans l'image d'entrée *im_in*.

La formule de calcul est tout simplement:

```
pixel(im_out)=pow(pixel(im_in),n)
```

L'image de sortie est de type Float.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Pour les graphes, le graphe de sortie est construit avec la puissance nième des valeurs de noeuds.

Paramètres

- *n* est un réel quelconque correspondant à la valeur de puissance.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *im_out*: une image de Floats ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Elève au carré l'image tangram.pan.

```
ppow 2 tangram.pan a.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PPow( const Img2duc &im_in, Img2duc &im_out, double n );
```

Auteur: Régis Clouard

ppowerlawtransform

Transformation des niveaux de gris par une loi de puissance.

Synopsis

ppowerlawtransform *gamma min max* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ppowerlawtransform** étale ou compresse les niveaux de gris selon une loi de transformation de puissance d'ordre *gamma*. Cette transformation est aussi connue sous le nom de correction gamma.

L'effet d'une transformation selon une loi de puissance est de plaquer une faible bande de niveaux de gris sombre de l'image initiale sur une large bande bande de niveaux de gris sombre de l'image de sortie gamma < 1, et le contraire quand gamma >1.

La transformation selon la loi de puissance d'un pixel 'p' prend la forme :

```
im_out[p]=(c * (im_in[p]-smin)^gamma) + min;
c=(max-min) / (smax-smin)
```

où smin et smax sont les valeurs minimale et maximale de l'image d'entrée et c est un facteur de normalisation pour l'étalement des valeurs de sortie entre *min* et *max*.

Pour les images couleur et multispectrales, la transformation utilise l'approche vectorielle : le min et le max sont calculés sur toutes les bandes et chaque bande est modifiée avec la même transformation.

Paramètres

- *gamma* est un réel positif. Il spécifie le degré de la transformation. Les valeurs < 1 compressent les niveaux de gris sombres et étalent les niveaux de gris clairs. Les valeurs >1 étalent les niveaux de gris clairs et compressent les niveaux de gris sombres. Une valeur de gamma=1 conduit à une transformation linéaire. Des valeurs typiques sont 0.04, ... , 0.4, 1, 1.5, ... 25.0.

- *min* et *max* spécifient les bornes des valeurs de pixels de l'image de sortie. Ils sont relatifs au type de l'image d'entrée.
  Note: si *min* < *max* alors min et max sont affectés rspectivement avec les valeurs minimum and maximum possibles pour le type de l'image d'entrée (ex: 0 et 255 pour des images de Uchar).

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image avec les mêmes propriétés que l'image d'entrée *im_in*.

Résultat

Retourne SUCCESS ou FAILURE en cas de mauvais paramétrage.

Exemples

Cette séquence applique successivement une transformation gamma puis la transformation inverse. L'image résultante b. est donc (presque) ègale à tangram.pan (à cause des arrondis de calcul):

```
ppowerlawtransform 2 0 255 tangram.pan a.pan
ppowerlawtransform 0.5 28 165 a.pan b.pan
```

Applique une transformation logarithmique pour créer l'image a.pan et utilise les valeurs extrémales possibles du type pour les bornes des valeurs de pixels de sortie :

```
ppowerlawtransform 0.4 1 -1 tangram.pan a.pan
```

Voir aussi

Transformation de la LUT, plineartransform, plogtransform

Prototype C++

```
Errc PPowerLawTransform( const Img2duc &im_in, const Img2duc
&im_out, float gamma, float min, float max );
```

Auteur: Régis Clouard

pppm2pan

Conversion d'un fichier de format PPM (Portable PixMap), PGM (Portable GrayMap) ou PBM (Portable BitMap) en un fichier Pandore.

Synopsis

pppm2pan *im_in* [*im_out*|-]

Description

L'opérateur **pppm2pan** permet de transformer une image de format PPM au format Pandore.

Un fichier PPM consiste en 2 parties, un entête et les données:

- L'entête consiste en 3 parties délimitées par un retour chariot ou par une fin de ligne (bien que la norme ne réclame que des espaces):
  - La première ligne contient le magic number.
    - P1: pour une image binaire avec les données en ASCII.
    - P2: pour une image en niveaux de gris avec les données en ASCII.
    - P4: pour une image binaire avec les données en binaire.
    - P5: pour une image en niveaux de gris avec les données en binaire.
    - P3: pour une image couleur avec les données en ASCII.
    - P6: pour une image couleur avec les données en binaire.
  - La seconde ligne donne le nombre de colonnes puis le nombre de lignes en ASCII.
  - La dernière partie donne le nombre de couleurs maximales utilisées.

  Chaque ligne peut contenir un commentaire introduit par #.

- Le format des données dépend du magic number. Dans le cas: Attention une ligne ne peut contenir plus de 70 caractères.
  - P2: les pixels sont codés en ASCII sous la forme d'un nombre entier 0 (blanc) ou 1 (noir)
  - P2: les pixels sont codés en ASCII sous la forme d'un nombre entier (le niveau de gris).
  - P4: les pixels sont codés en binaire (o : blanc, 1 : noir).
  - P5: les pixels sont codés en binaire sous la forme d'un nombre entier (le niveau de gris).
  - P3: les pixels sont codés en ASCII sous la forme de trois nombres consécutifs (Rouge Vert Bleu).
  - P6: les pixels sont codés en binaire sous la forme de 3 octets consécutifs (Rouge Vert Bleu).

Exemple:

```
P2
#Image en niveaux de gris
#de 5 colones et de 4 lignes.
5 4
255
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

Astuce: Le format ppm permet de construire facilement une image à partir d'une matrice de valeurs codées par exemple en ascii. Il suffit de rajouter un entete ppm au fichier de données.

Entrées

- *im_in*: un fichier PPM, PGM ou PBM.

Sorties

- *im_out*: une image 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Convertit une image ppm en image pandore, puis l'affiche

```
pppm2pan image.ppm image.pan
pvisu image.pan
```

Voir aussi

Conversion, ppan2ppm

Prototype C++

```
Errc PPPM2Pan( const char* filename, Pobject **obj_out );
```

Auteur: Régis Clouard

pprewitt

Module du gradient de Prewitt.

Synopsis

pprewitt [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pprewitt** permet d'approximer le calcul de l'amplitude du gradient de l'image *im_out*.

L'algorithme consiste à convoluer l'image par le masque de Prewitt:

```
|+1 +1 +1|
|+0 +0 +0|
|-1 -1 -1|
```

En fait ici, le masque est orienté successivement dans les quatre directions: 0, 45, 90, 135 degrés et c'est la valeur maximale qui est choisie comme amplitude.

L'image de sortie *im_out* est de même type que l'image d'entrée *im_in*.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image du même type que l'image im_in.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Détection de contours dans l'image tangram.pan :

```
pprewitt tangram.pan b.pan
pbinarization 40 1e30 b.pan out.pan
```

Voir aussi

Détection de contours

Prototype C++

```
Errc PPrewitt( const Img2duc &im_in, Img2duc &ima );
```

Auteur: Régis Clouard

pproperty

Retourne la valeur d'une propriété d'un objet Pandore.

Synopsis

pproperty *property_index* [*im_in*|-]

Description

L'opérateur **pproperty** permet de retourner la valeur de la propriété *property_index* de l'objet Pandore *im_in*. La valeur est ensuite récupérable (et affichable) par l'opérateur *pstatus*.

Paramètres

- *property_index* est un entier de l'intervalle [0-4] avec la convention suivante:
  - 0 : Le nombre de colonnes d'une image, carte de régions ou graphe.
  - 1 : Le nombre de lignes d'une image, carte de régions ou graphe.
  - 2 : Le nombre de plans d'une image, carte de régions ou graphe.
  - 3 : Le nombre de bandes d'une image ou
    le nombre de labels dans une carte de région ou
    le nombre de noeuds pour les graphes (size).
  - 4 : Le numéro de l'espace couleur d'une image couleur parmi:
    [0: RGB; 1: XYZ; 2: LUV; 3: LAB; 4: HSL; 5: AST; 6: I1I2I3; 7: LCH; 8: WRY; 9: RNGNBN; 10: YCBCR; 11: YCH1CH; 12: YIQ; 13: YUV].

Entrées

- *im_in*: un fichier Pandore.

Résultat

Retourne la valeur de la propriété sélectionnée ou FAILURE.

Exemples

- Unix/Linux/MACOS : Construit une nouvelle image avec les mêmes dimensions que l'image tangram.pan qui contient un disque blanc de rayon 10 pixels :

```
pproperty 0 tangram.pan; w=‘pstatus‘
pproperty 1 tangram.pan; h=‘pstatus‘
pshapedesign $w $h 0 1 50 50 a.pan
```

- MsDos : Construit une nouvelle image avec les mêmes dimensions que l'image tangram.pan qui

contient un disque blanc de rayon 10 pixels :

```
pproperty 0 tangram.pan
call pstatus
call pset w
pproperty 1 tangram.pan
call pstatus
call pset h
pshapedesign $w $h 0 1 50 50 a.pan
```

Voir aussi

Information

Prototype C++

```
Errc PProperty( const Img2duc &img, int property_index );
```

Auteur: Régis Clouard

ppsnr

Calcul du rapport signal sur bruit en crête.

Synopsis

ppsnr *max* [*im1_in*|-] [*im2_in*|-]

Description

L'opérateur **ppsnr** mesure rapport signal sur bruit en crête entre l'image initiale *im_in1* et sa version restaurée ou améliorée *im_in2*.
Le PSNR (Peak Signal to Noise Ratio) est le rapport entre la puissance maximale du signal et la puissance du bruit qui affecte la fidelité de sa représentation. Il est défini par l'erreur quadratique moyenne (MSE) entre les deux images d'entrée où *im1_in* est l'image initiale et *im2_in* est la version restaurée ou améliorée de *im1_in*.
En conséquence, plus le PSNR est élevé, meilleure est le signal et donc le traitement de restauration ou d'amélioration.

Parce que les valeurs peuvent occuper une très grande plage de valeur, le PSNR est exprimée en décibel (dB). Les valeurs typiques pour le PSNR d'une compression d'image sont comprises entre 30 and 40 dB.

Le PSNR est défini comme suit :

$$PSNR = 10 * \log_{10}((max*max)/MSE);$$
$$\text{with } MSE=sum((im\_in1-im\_in2)^2)/N$$

où *max* est la valeur de pixel maximale et N le nombre total de pixel de l'image d'entrée. Si *max*=-1 alors max=Max(ims_in1)-Min(im_in1)).

Les images d'entrée *im_in1* et *im_in2* doivent avoir les mêmes dimensions et le même type.

Pour les images couleur et multispectrales, la définition du PSNR est la même sauf que l'erreur quadratique moyenne est la somme des erreurs sur toutes les bands.

Param&eagrave;tre

- *max* est la valeur de pixel maximale. Typiquement, max=255 pour les images de Uchar.
  Si *max*=-1 alors max=Max(ims_in1)-Min(im_in1)).

Entrées

- *im_in1*: une image.
- *im_in2*: une image (une version restaurée ou améliorée de im_in1).

Résultat

Retourne une valeur réelle positive exprimée en décibel (dB).
(Utiliser `pstatus` pour récupérer cette valeur).

Exemples

Ajoute un bruit gaussien de moyenne 0 et d'écart type 1.5 à l'image tangram.pan et puis calcule le PSNR pou un filtre moyenneur :

```
paddnoise 1 0 1.5 tangram.pan a.pan
pmeanfilter 2 a.pan i1.pan
ppsnr 255 tangram.pan i1.pan
pstatus
```

Voir aussi

Evaluation, pmse, psnr

Prototype C++

```
Errc PPSNR( const Img2duc &im_in1, const Img2duc &im_in2, Float max
);
```

Auteur: Régis Clouard

pqmf

Génération d'un filtre QMF pour la transformée en ondelette.

Synopsis

pqmf *name order* [*col_out*|-]

Description

L'opérateur **pqmf** (Quadratic Mirror Filter) permet de créer un fichier collection contenant les informations sur un filtre nécessaire à l'utilisation de pdwt. Ces informations sont le nom et l'ensemble des coefficients du filtre passe-bas utilisé pour le calcul de l'approximation lors de la décomposition en ondelettes dyadiques d'une image. Cela donne aussi, implicitement, la longueur du filtre.

Paramètres

- *name* spécifie le nom du filtre parmi les 7 filtres disponibles.
- *order* specifie l'ordre du filtre.
  - haar: 1
  - beylkin: 1
  - coiflet: 1, 2, 3, 4 ou 5
  - daubechies: 4, 6, 8, 10, 12, 14, 16, 18 ou 20
  - symmlet: 4, 5, 6, 7, 8, 9 ou 10
  - vaidyanathan: 1
  - battle: 1, 3 ou 5

Sortie

- *col_out*: une collection qui contient les coefficients du filtre choisi.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit une image synthétique avec un carré pour illustrer le phénomène de Gibbs an analyse par ondelettes:

```
pshapedesign 256 256 0 2 150 150 a.pan
pqmf daubechies 4 b.pan
pdwt 1 a.pan b.pan c.pan
psplitimage c.pan d1.pan d2.pan d3.pan d4.pan
pthresholding 20 400 d2.pan e2.pan
pthresholding 20 400 d3.pan e3.pan
pthresholding 20 400 d4.pan e4.pan
pmergeimages d1.pan e2.pan e3.pan e4.pan f.pan
pidwt 1 f.pan b.pan out.pan
```

Voir aussi

Domaine Fréquentiel, dwt, idwt

Prototype C++

```
Errc PQmf( const char *name, const char *order, Collection& col_out
);
```

Auteur: Ludovic Soltys

pquadrangle2rectangle

Corrige la distortion géométrique du contenu d'une image pour passer de la représentation d'un quadrilatère à celle d'un rectangle.

Synopsis

pquadrangle2rectangle [*-m mask*] [*im1_in*|-] [*im2_in*|-] [*im_out*|-]

Description

L'opérateur **pquadrangle2rectangle** permet de redresser la partie de l'image *im2_in* qui est incluse dans le quadrilatère repérée par les 4 coins donnés dans l'image *im1_in*.

Entrées

- *im1_in*: une image contenant les 4 coins du quadrilatère.
- *im2_in*: une image couleur ou niveau de gris à redresser.
- *im_out*: une image du même type que l'image *im2_in*.

Résultat

SUCCESS ou FAILURE si l'image n'a pas pu être redressée.

Exemples

Détection des bords du tableau blanc :

```
pshen 1.3 whiteboard.pan a.pan
pbinarization 2 255 a.pan bin.pan
phoughlines 10 0 360 a.pan lines.pan
pgetquadrangle lines.pan a.pan b.pan
pquadrangle2rectangle b.pan whiteboard.pan result.pan
```

Voir aussi

Reconstruction

Prototype C++

```
Errc PQuadrangle2Rectangle( const Img2duc &im1_in, const Img2duc
&im2_in, Img2duc im_out );
```

Reference

Z. Zhang, and L. He, "*Whiteboard Scanning and Image Enhancement*", Digital Signal Processing, Vol.17, No.2, pages 414-432, 2007.

Auteur: Régis Clouard

pranksegmentationalgorithms

Classement d'algorithmes de segmentation à partir de mesures de dissimilarité entre des résultats de segmentation et des segmentations de référence.

Synopsis

pranksegmentationalgorithms *[-v] acceptable_error1 acceptable_error2 acceptable_error3 acceptable_error4 acceptable_error5 col_in** [*col_out1*|-] [*col_out2*|-]

Description

L'opérateur **pranksegmentationalgorithms** permet de classer plusieurs algorithmes de segmentation d'images selon leurs performances. Les performances sont évaluées par comparaison des résultats des algorithmes sur des images test avec des segmentations de référence fournies par des experts.

Les performances sont calculées à partir de mesures de dissimilarité entre les résultats de segmentation et les segmentations de référence correspondantes. Cinq indicateurs de dissimilarité sont évalués, et à chaque fois, deux mesures sont calculées avec une valeur entre 0 et 1 :

- **Indicateur 1** : La **précision de la détection** : Les deux erreurs sont :
  - L'erreur de rappel qui rend compte de la proportion de faux négatifs.
  - L'erreur de précision qui rend compte de la proportion de faux positifs.
- **Indicateur 2** : La **cohérence de la fragmentation** : Les deux erreurs sont :
  - L'erreur de sous-segmentation qui rend compte de la proportion de régions agglomérées par segment.
  - L'erreur de sur-segmentation qui rend compte de la proportion de fragmentation des régions en plusieurs segments.
- **Indicateur 3** : La **localisation des frontières** : Les deux erreurs sont :
  - L'erreur de déficit de pixels qui rend compte de la proportion de pixels non détectés dans les régions détectées.
  - L'erreur d'excès de pixels qui rend compte de la proportion de pixels erronés ajoutés aux régions détectées.
- **Indicateur 4** : Le **respect de la forme** : Les deux erreurs sont :
  - L'erreur de forme due à l'omission de surface des régions.
  - L'erreur de forme due à l'ajout de surface aux régions.
- **Indicateur 5** : La **préservation de la topologie** : Les deux erreurs sont :
  - L'erreur d'ajout de trou qui rend compte de la proportion de faux trous détectés.
  - L'erreur de suppression de trou qui rend compte de la proportion de trous non détectés.

Pour chaque indicateur *i*, il est nécessaire de préciser l'erreur qu'il faut considérer comme la plus acceptable parmi les deux possibles par l'intermédiaire du paramètre $acceptable\_error_i$ (voir la section "Paramètres").

Le résultat est stocké dans deux collections. La première collection *col_out1* contient les valeurs pour les cinq indicateurs de performances ci-dessus. La deuxième collection *col_out2* contient les rangs de chacun des algorithmes.

Paramètres

- *-v* : mode verbeux
- *acceptable_error1*, *acceptable_error2*, *acceptable_error3*, *acceptable_error4*, *acceptable_error5* : permet d'indiquer pour chaque indicateur quelle est l'erreur qu'il faut considérer comme la plus acceptable. Les valeurs sont comprises entre 1 et 8:
  1. Les deux erreurs sont acceptables (pas de pénalisation).
  2. Les deux erreurs sont indésirables.
  3. Préférer l'erreur1 à l'erreur2.
  4. Préférer l'erreur2 à l'erreur1.
  5. Ne pas pénaliser l'erreur1.
  6. Ne pas pénaliser l'erreur2.
  7. Prohiber l'erreur1.
  8. Prohiber l'erreur2.

Entrées

- *col_in**: une liste de collections dont chacune contient les 10 erreurs de segmentation d'un algorithme (calculées par l'opérateur passesssegmentationalgorithm).

Sorties

- *col_out1*: une collection contenant les valeurs d'indicateurs pour chaque algorithme.
- *col_out2*: une collection contenant les rangs de chaque algorithme.

Résultat

Retourne SUCCESS ou FAILURE en cas de problème.

Exemples

Classement de deux algorithmes à, partir de leurs résultats de segmentation&nbps;:

```
passesssegmentationalgorithm 0 0.5 resultimages/algo001 groundtruths detail_errors_algo1.pan total_errors_algo1.pan
passesssegmentationalgorithm 0 0.5 resultimages/algo002 groundtruths detail_errors_algo2.pan total_errors_algo2.pan
pranksegmentationalgorithms 4 6 3 1 1 total_errors_algo1.pan total_errors_algo1.pan indicators.pan rank.pan
```

Voir aussi

Evaluation, passessdetectionaccuracy, passessfragmentationconsistency, passessboundaryprecision, passessshapefidelity, passesstopologypreservation, passesssegmentationalgorithm, pranksegmentationalgorithmsfromfolders

Prototype C++

```
Errc PRankAlgorithmsSegmentation(Pobject ** cols, int nbAlgorithms,
Collection & cold1, Collection & cold2, int acceptable_error1, int
acceptable_error2, int acceptable_error3, int acceptable_error4, int
acceptable_error5);
```

Auteur: Régis Clouard

pranksegmentationalgorithmsfromfolders

Classement d'algorithmes de segmentation à partir de mesures de dissimilarité entre des résultats de segmentation et des segmentations de référence (complet).

Synopsis

pranksegmentationalgorithmsfromfolders *matching_algorithm_id matching_threshold acceptable_error1 acceptable_error2 acceptable_error3 acceptable_error4 acceptable_error5 segmentation_result_path reference_segmentation_path* [*col_out1*|-] [*col_out2*|-]

Description

L'opérateur **pranksegmentationalgorithmsfromfolders** permet de classer plusieurs algorithmes de segmentation d'images selon leurs performances. Les performances sont évaluées par comparaison des résultats des algorithmes sur des images test avec des segmentations de référence fournies par des experts.

Les performances sont calculées à partir de mesures de dissimilarité entre les résultats de segmentation et les segmentations de référence correspondantes. Cinq indicateurs de dissimilarité sont évalués, et à chaque fois, deux mesures sont calculées avec une valeur entre 0 et 1 :

- **Indicateur 1** : La **précision de la détection** : Les deux erreurs sont :
  - L'erreur de rappel qui rend compte de la proportion de faux négatifs.
  - L'erreur de précision qui rend compte de la proportion de faux positifs.
- **Indicateur 2** : La **cohérence de la fragmentation** : Les deux erreurs sont :
  - L'erreur de sous-segmentation qui rend compte de la proportion de régions agglomérées par segment.
  - L'erreur de sur-segmentation qui rend compte de la proportion de fragmentation des régions en plusieurs segments.
- **Indicateur 3** : La **localisation des frontières** : Les deux erreurs sont :
  - L'erreur de déficit de pixels qui rend compte de la proportion de pixels non détectés dans les régions détectées.
  - L'erreur d'excès de pixels qui rend compte de la proportion de pixels erronés ajoutés aux régions détectées.
- **Indicateur 4** : Le **respect de la forme** : Les deux erreurs sont :
  - L'erreur de forme due à l'omission de surface des régions.
  - L'erreur de forme due à l'ajout de surface aux régions.
- **Indicateur 5** : La **préservation de la topologie** : Les deux erreurs sont :
  - L'erreur d'ajout de trou qui rend compte de la proportion de faux trous détectés.
  - L'erreur de suppression de trou qui rend compte de la proportion de trous non détectés.

Le dossier d'entrée est supposé contenir autant de sous-dossiers qu'il y a d'algorithmes à classer. De même, le dossier des segmentations de référence contient autant de sous-dossiers qu'il a d'expertise pour les images test. Chaque sous-dosier est organisé de la même manière avec les mêmes noms d'images.

Pour chaque indicateur *i*, il est nécessaire de préciser l'erreur qu'il faut considérer comme la plus acceptable parmi les deux possibles par l'intermédiaire du paramètre $acceptable\_error_i$ (voir la section "Paramètres").

Le résultat est stocké dans deux collections. La première collection *col_out1* contient les valeurs pour les cinq indicateurs de performances ci-dessus. La deuxième collection *col_out2* contient les rangs de chacun des algorithmes.

Paramètres

- *-v* : mode verbeux
- *matching_algorithm_id* : spécifie le numéro de l'algorithme de mise en correspondance à utiliser :
  - 0 : pour une correspondance de type 1-n et n-1. Un segment d'un résultat de segmentation peut regrouper plusieurs régions de la référence (sous-segmentation), et une région de la référence peut être découper en plusieurs segments d'un résultat de segmentation (sur-segmentation). Toutefois, un segment ou une région ne peut participer à la fois à une sur-segmentation et à une sous-segmentation.
  - 1 : pour une correspondance de type 1-1. Un segment de la segmentation ne peut être mis en correspondance qu'avec au plus une région de la référence, et une région de la référence ne peut être mise en correspondance qu'avec au plus un segment de la segmentation.
- *acceptable_error1*, *acceptable_error2*, *acceptable_error3*, *acceptable_error4*, *acceptable_error5* : permet d'indiquer pour chaque indicateur quelle est l'erreur qu'il faut considérer comme la plus acceptable. Les valeurs sont comprises entre 1 et 8:
  1. Les deux erreurs sont acceptables (pas de pénalisation).
  2. Les deux erreurs sont indésirables.
  3. Préférer l'erreur1 à l'erreur2.
  4. Préférer l'erreur2 à l'erreur1.
  5. Ne pas pénaliser l'erreur1.
  6. Ne pas pénaliser l'erreur2.
  7. Prohiber l'erreur1.
  8. Prohiber l'erreur2.
- *segmentation_result_path* : le chemin vers le dossier des résultats de segmentation de l'algorithme. Ce dossier peut être organisé en sous-dossiers.
- *reference_segmentation_path* : le chemin vers le dossier des segmentations de référence. Le dossier est divisé en autant de sous-dossiers qu'il y a d'expertises disponibles sur les images. Chaque sous-dossier d'expertise est organisé de la même façon qu'un sous-dossier du dossier *segmentation_result_path* avec les mêmes noms d'image.

Sorties

- *col_out1*: une collection contenant les valeurs d'indicateurs pour chaque algorithme.
- *col_out2*: une collection contenant les rangs de chaque algorithme.

Résultat

Retourne SUCCESS ou FAILURE en cas de problème.

Exemples

Classement des algorithmes à, partir de leurs résultats de segmentation&nbps; stok&es dans des sous-dossiers du dossier ’resultimages’:

```
pranksegmentationalgorithmsfromfolders 0 0.5 4 6 3 1 1 resultimages groundtruths indicators.pan rank.pan
```

Voir aussi

Evaluation, passessdetectionaccuracy, passessfragmentationconsistency, passessboundaryprecision, passessshapefidelity, passesstopologypreservation, passesssegmentationalgorithm, pranksegmentationalgorithms

Prototype C++

```
Errc PRankSegmentationAlgorithmsFromFolders( int
matching_algorithm_id, float matching_threshold, int
acceptable_error1, int acceptable_error2, int acceptable_error3, int
acceptable_error4, int acceptable_error5, std::string
segmentation_result_path, std::string ground_truth_path, Collection
& col_out1, Collection & col_out2 );
```

Auteur: Régis Clouard

prankthresholding

Seuillage d'une image selon le rang des valeurs de pixels.

Synopsis

prankthresholding *seuilb seuilh* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prankthresholding** consiste à mettre à 0, toutes les valeurs de pixel dont le rang est inférieur strictement au *seuilb* ou supérieur strictement au *seuilh* et à conserver les valeurs de pixel comprises entre ces deux seuils.

```
if rank(im_in[p]) ≥ low and rang(im_in[p]) ≤ high
then im_out[p] = im_in[p];
else im_out[p] = 0;
```

Si *high* est inférieur à *low* alors **prankthresholding** effectue le seuillage à l'envers:

```
if rank(im_in[p]) < high or rank(im_in[p]) > low
then im_out[p] = im_in[p];
else im_out[p] = 0;
```

L'image de sortie *im_out* est du même type que celle d'entrée *im_in*.

Paramètres

- Le *seuilb* et le *seuilb* sont des valeurs entières correspondant à des rangs.

  Astuce: Si *seuilh* est supérieur à la valeur maximale du type des pixels alors c'est la valeur maximale qui est utilisée. (ex: 255 pour Img2duc, +2147483648 pour Img2dsl).

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *im_out*: une image de niveaux de gris.

Résultat

Retourne SUCCESS ou FAILURE en cas de mauvais paramétrage.

Exemples

Selection des pixels appartenant aux 18 plus petites values:

```
prankthresholding 0 17 examples/tangram.pan out.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PRankThresholding( const Img2duc &im_in, Img2duc &im_out, Uchar seuilb, Uchar seuilh );
```

Auteur: Régis Clouard

pras2pan

Conversion d'une image Rasterfile en une image Pandore.

Synopsis

pras2pan *im_in* [*im_out*|-]

Description

L'opérateur **pras2pan** permet de convertir une image de type Sun Rasterfile en un fichier au format Pandore.

Le type de l'image de sortie *im_out* dépend du type de l'image d'entrée *im_in*, mais il s'agit toujours d'une image 2D couleur.

Entrées

- *im_in*: une image Sun Rasterfile.

Sorties

- *im_out*: une image Pandore 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit une image PPM en image Pandore:

```
pppm2pan image.ppm image.pan
```

Voir aussi

Conversion

Prototype C++

```
Errc PRas2Pan( const Char* filename, Pobject **objout );
```

Auteur: Régis Clouard

praw2pan

Conversion d'un fichier image sans entête en un fichier Pandore.

Synopsis

praw2pan *bytes ncol nlig ndep color im_in* [*im_out*|-]

Description

L'opérateur **praw2pan** permet de transformer une image d'un fichier sans entête en image au format Pandore.

Le paramètre *bytes* détermine la taille d'un pixel en octet et ce qui va déterminer le type de l'image de sortie. Les données sont données en binaire les unes derrière les autres dans le sens causal.

Dans le cas d'une image couleur, le paramètre *color* permet d'indiquer s'il s'agirt d'une niamge en niveaux de gris ou en couleur et le forme de coadge des couleurs (par vecteur rgb pour chaque pxiel ou par bandes).

Attention: Si l'image vient d'un fichier fait sur une machine utilisant une autre technologie de microprocesseurs (Inter / Motorola), il y a inversion de valeurs de pixels. Dans ce cas, et par pure convention, la taille d'un octet est donnée en valeur négative (ex: -1 pour une image d'octets, -2 pour une image de short).

Astuce: **praw2pan** élimine tout éventuel entête du fichier raw en récupérant le nombre de données spécifié par les paramètres depuis la fin du fichier.

Paramètres

- *bytes* spécifie le nombre d'octets utilisés pour coder un pixel. *Par convention, une valeur négative indique qu'en plus il faut faire une inversion des bytes (LSB <->MSB).*
  C'est aussi ce paramètre qui détermine le type de l'image de sortie:
  - bytes=1 (ou -1) crée une image de Uchar.
  - bytes=2 (ou -2) crée une image de Slong.
  - bytes=3 (ou -3) crée une image de Slong.
  - bytes=4 (ou -4) crée une image de Slong.
  - bytes=6 (ou -6) crée une image de Float (pure convention).
  - bytes=8 (ou -8) crée une image de Double (pure convention).
- *ncol*, *nlig* et *ndep* spécifient la taille de l'image. Le type de l'image de sortie dépend de la valeur de ces paramètres. Si les trois paramètres sont strictement positifs alors l'image sera 3D, si *ndep*=0 alors l'image sera 2D, si en plus *nrow*=0 alors l'image sera 1D.
- *color* si color=0 alors il s'agit d'une image en niveaux d egris. Si color = 1 alors les images couleur son codées par vecteur rgb pour chaque pixel, sinon elles sont codées par bande.

Entrées

- *im_in*: un fichier binaire.

Sorties

- *im_out*: une image Pandore.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Conversion d’une image couleur 2D 1024x1024 pixels codées par bande.

```
praw2pan 1 1024 1024 0 0 image.raw image.pan
```

Voir aussi

Conversion, ppan2raw

Prototype C++

```
Errc PRaw2Pan( char *filename, Img2duc &imd, int bytes, int color );
```

Auteur: Régis Clouard

prds

Construction d'une image stéréogramme type Random Dot Stereogram.

Synopsis

prds [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prds** construit une image Random Dot Stereogram à partir de l'image de profondeur *im_in*. Un stéréogramme est une image dans laquelle une information stéréoscopique est codée. Pour regarder un stéréogramme, il faut considérer que le plan focal est derrière l'image.

L'image de profondeur est une image de niveaux de gris telle que chaque pixel code la valeur de profondeur du point correspond dans la scène.

L'image finale est apparemment une image de bruit mais l'objet 3D est codé à l'intérieur.

Entrées

- *im_in*: une image 2D d'octets (Img2duc).

Sorties

- *im_out*: une image 2D d'octets (Img2duc).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcul un RDS à partir de l'image de profondeur depth.pan :

```
prds depth.pan a.pan
```

Voir aussi

Exotique

Prototype C++

```
Errc PRds( const Img2duc &imp, Img2duc &im_out );
```

Auteur: Régis Clouard

prectangularityselection

Sélection de régions sur leur valeur de rectangularité.

Synopsis

prectangularityselection *relation seuil* [*-m mask*] [*rg_in*|*-*] [*rg_out*|*-*]

Description

L'opérateur **prectangularityselection** permet de sélectionner les régions sur leur degré de rectangularité. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

La rectangularité est égale au rapport de la surface de la région sur la surface du rectangle exinscrit. Le rectangle exinscrit est pris comme celui qui épouse le mieux la forme de la région, c'est à dire celui qui donne un coefficient de rectangularité maximum parmi toutes les rotations possibles du rectangle.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur réelle [0..1] qui correspond au facteur de rectangularité.
  Cette valeur est égale à 1 pour un rectangle.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Exemples

Sélectionne les régions avec un degré de rectangularité > 10:

```
prectangularityselection 1 10 in.pn rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PRectangularitySelection( const Reg2d &rg_in, Reg2d &rg_out,
int relation, float seuil );
```

Auteur: Régis Clouard

pregionalmaxima

Localisation des points constituant un maximum régional.

Synopsis

pregionalmaxima *lenght* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pregionalmaxima** construit une image avec les points maxima régionaux selon la taille *length* de la région.

Un point est maximal s'il ne possède aucun voisin plus petit que lui dans une région de demi-taille *length* autour de lui. Les pixels de valeur maximale sont mis à 255 dans l'image de sortie *im_out* les autres sont mis à 0.

Paramètres

- *length* définit la notion de distance maximale entre deux maxima.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *im_out*: une image binaire.

Exemples

Localise les maxima éloignés d'au moins 10 pixels des autres dans l'image tangram.pan:

```
pregionalmaxima 10 examples/tangram.pan a.pan
```

Résultat

Retourne SUCCESS ou FAILURE.

Voir aussi

Caractérisation image, pregionalminima

Prototype C++

```
Errc PRegionalMaxima( const Img2duc &im_in, Img2duc &im_out, int
length );
```

Auteur: Régis Clouard

pregionalminima

Localisation des points constituant un minimum régional.

Synopsis

pregionalminima *length* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pregionalminima** construit une image avec les points constituant un minima régional. La taille de la région est définie par le paramètre *length* donné en pixels.

Un point est minimal s'il ne possède aucun voisin plus petit que lui à une distance *length/2* autour de lui. Les pixels de valeur minimale sont mis à 255 dans l'image de sortie *im_out* les autres sont mis à 0.

Paramètres

- *length* définit la notion de distance minimale entre deux minima. Il est donné en pixels.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *im_out*: une image binaire.

Exemples

Localise les minima éloignés d'au moins 10 pixels des autres dans l'image tangram.pan:

```
pregionalminima 10 examples/tangram.pan a.pan
```

Résultat

Retourne SUCCESS ou FAILURE.

Voir aussi

Caractérisation image, pregionalmaxima

Prototype C++

```
Errc PRegionalMinima( const Img2duc &im_in, Img2duc &im_out, int
length );
```

Auteur: Régis Clouard

pregionarea

Calcul de la surface des régions.

Synopsis

pregionarea *attr* [*-m mask*] [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregionarea** crée une collection *col_out* contenant un tableau nommé *attr* de Ulong dans lequel le ième correspond à la surface de la région de label i+1.

La valeur de surface est calculée en nombre de pixels inclus dans la région et sur la frontière. Pour les creux entre deux pixels, on ajoute la moitié de la surface manquante. Sur l'exemple ci-dessous, la surface est de 10=8+4*0,5.

```
 xx
xxxx
 xx
```

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de surface dans la collection.

Entrées

- *rg_in*: une cartes de régions 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la surface des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionarea area b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionArea( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregioncompactness

Calcul de la compacité des régions.

Synopsis

pregioncompactness *attr* [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregioncompactness** crée une collection *col_out* contenant un tableau nommé *attr* de float dans lequel le ième correspond è la compacité de la région de label i+1.

La compacité est un facteur de circularité qui vaut 1 lorsque la région est un cercle, et diminue à mesure que le contour est très découpé ou que la région est allongée.

Elle est calculée par la formule:

```
compacité. (4*PI*surface) / (perimetre*perimetre)
```

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de compacité dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la compacité des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregioncompactness compactness b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionCompactness( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregionconvexity

Calcul de la convexité des régions.

Synopsis

pregionconvexity *attr* [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregionconvexity** crée une collection *col_out* contenant un tableau nommé *attr* de Ulong dans lequel le ième correspond à la convexité de la région de label i+1.

Le calcul de la mesure est fait par:

```
convexité = Surface de la région / Surface de l'enveloppe convexe.
```

Une région fortement convexe (ie, qui épouse parfaitement son enveloppe convexe) a une valeur de convexité = 1.
Une région faiblement convexe a une valeur de convexité << 1.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de convexité dans la collection.

Entrées

- *rg_in*: une carte de régions2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la convexité des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionconvexity convexity b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionConvexity( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregiondensity

Calcul de la densité des régions.

Synopsis

pregiondensity *attr* [*rg_in*|-] [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pregiondensity** crée une collection *col_out* contenant un tableau nommé *attr* de float dans lequel le ième correspond au facteur de densité de la région de label i+1.

La densité est le rapport entre le number de pixels dans la régions données par *im_in* et la surface de la région donnée par l'entrée *rg_in*.

```
densite = nombre de pxiels / surface.
```

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de densité dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.
- *im_in*: une image 2D d'octets.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la densité des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregiondensity density b.pan tangram.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionDensity( const Reg2d &rg_in, const Img2duc &ims;
Collection &cold, const std::string &attr );
```

Auteur: Régis Clouard

pregionheight

Calcul de la profondeur des régions.

Synopsis

pregionheight *attr* [*-m mask*] [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregionheight** crée une collection *col_out* contenant un tableau nommé *attr* de Ulong dans lequel le ième correspond à la profondeur de la région de label i+1.

La valeur de profondeur est calculée en nombre de pixels

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de profondeur dans la collection.

Entrées

- *rg_in*: une cartes de régions.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la profondeur des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionheight area b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionHeight( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregioneccentricity

Calcul de l'excentricité des régions.

Synopsis

pregioneccentricity *attr* [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregioneccentricite** crée une collection *col_out* contenant un tableau *attr* de float dans lequel le ième correspond à l'excentricité de la région de label i+1.

L'excentricité est calculée par:

$$\text{excentricite} = \frac{(Mxx+Myy-\sqrt{(Mxx-Myy)*(Mxx-Myy)+4*Mxy*Mxy})}{(Mxx+Myy+\sqrt{(Mxx-Myy)*(Mxx-Myy)+4*Mxy*Mxy})}$$

Elle correspond au rapport de la longueur du petit axe sur celle du grand axe d'une région. Le résultat est une valeur réelle entre 0 et 1, avec 1 pour un carré.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs d'excentricité dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche l'excentricité des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregioneccentricity eccentricity b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionEccentricity( const Reg2d &rg_in, Collection &cold,
const std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregionelongation

Calcul de l'élongation des régions.

Synopsis

pregionelongation *attr* [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregionelongation** crée une collection *col_out* contenant un tableau nommé *attr* de float dans lequel le ième correspond à l'élongation de la région de label i+1.

L'élongation est le rapport entre la largeur et la longueur du rectange exincrit des régions:

```
elongation = largeur(rectangle)/longueur(rectangle).
```

L'élongation est une valeur réelle entre [0..1] , elle est égale à 1 pour un carré ou un disque, et <<1 pour un objet oblongue.

Le rectangle exinscrit est calculé avec différentes orientations, et on conserve celui qui donne l'élongation minimale, en considérant que c'est celui qui épouse le mieux la forme.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs d'élongation dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche l'élongation des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionelongation elongation b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionElongation( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregionenergy

Calcul de l'énergie des régions.

Synopsis

pregionenergy *attr* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pregionenergy** crée une collection *col_out* contenant un tableau nommé *attr* de Double dans lequel le ième correspond à l'énergie calculée dans l'image *im_in* de la région numérotée i+1 dans *rg_in*.

La mesure d'énergie est faite selon la formule:

```
energie = SOMME{ im_in[p] * im_in[p]} / N
```

où N est le nombre de pixels de la région.

Plus une région est lumineuse plus sont énergie est forte.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs d'énergie dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.
- *im_in*: une image 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche l’énergie des régions obtenues par une simple binarisation de l’image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionenergy energy b.pan tangram.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionEnergy( const Reg2d &rg_in, const Img2duc &im_in,
Collection &cold, const std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregioneulernumber

Calcul du nombre d'Euler des régions.

Synopsis

pregioneulernumber *attr* [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregioneulernumber** crée une collection *col_out* contenant un tableau nommé *attr* de Long dans lequel le ième correspond au nombre d'Euler de la région de label i+1.

Le nombre d'Euler d'une région est:

```
euler = nombre de composante connexe - nombre de trou.
```

Pour une région, le nombre de composante connexe =1, puisqu'une région est définie comme étant une composante connexe.

En fait, ici le nombre d'Euler est calculé à partir d'une opération locale:

```
Soit X(R) le nombre de pattern 2x2
(r label de la région R, et 0 tout autre label):
0 0
0 r

Soit V(R) le nombre de pattern 2x2:
0 r
r r

alors Euler(R) = X(R) - V(R)
```

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs du nombre d'Euler dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche le nombre d'Euler des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregioneulernumber eulernumber b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionEulerNumber( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Régis Clouard

pregionheight

Calcul de la hauteur des régions.

Synopsis

pregionheight *attr* [*-m mask*] [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregionheight** crée une collection *col_out* contenant un tableau nommé *attr* de Ulong dans lequel le ième correspond à la hauteur de la région de label i+1.

La valeur de hauteur est calculée en nombre de pixels

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de hauteur dans la collection.

Entrées

- *rg_in*: une cartes de régions.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la hauteur des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionheight area b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionHeight( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregionmaximum

Calcul de la valeur maximale des régions.

Synopsis

pregionmaximum *attr* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pregionmaximum** crée une collection *col_out* contenant un tableau nommé *attr* de Double dans lequel le ième correspond au maximum calculé dans *im_in* de la région numérotée i+1 dans *rg_in*.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs maximales dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.
- *im_in*: une image de niveaux de gris 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la valeur de pixel maximale des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionmax max b.pan tangram.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionMaximum( const Reg2d &rg_in, const Img2duc &im_in,
Collection &cold, const std::string &attr );
```

Auteur: Régis Clouard

pregionmean

Calcul de la moyenne des régions.

Synopsis

pregionmean *attr* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pregionmean** crée une collection *col_out* contenant un tableau nommé *attr* de Double dans lequel le ième correspond à la moyenne calculée dans *im_in* de la région numérotée i+1 dans *rg_in*.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de moyenne dans la collection.

Entrées

- *rg_in*: les cartes de régions 2D.
- *im_in*: une image 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la valeur de pixel moyenne des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionmean mean b.pan tangram.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionMean( const Reg2d &rg_in, const Img2duc &im_in,
Collection &cold, const std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregionminimum

Calcul de la valeur minimale des régions.

Synopsis

pregionminimum *attr* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pregionminimum** crée une collection *col_out* contenant un tableau nommé *attr* de Double dans lequel le ième correspond au minimum calculée dans *im_in* de la région numérotée i+1 dans *rg_in*.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs minimales dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.
- *im_in*: une images de niveaux de gris 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la valeur de pixel minimale des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionminimum min b.pan tangram.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionMinimum( const Reg2d &rg_in, const Img2duc &im_in,
Collection &cold, const std::string &attr );
```

Auteur: Régis Clouard

pregionorientation

Calcul de l'orientation des régions.

Synopsis

pregionorientation *attr* [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregionorientation** crée une collection *col_out* contenant un tableau nommé *attr* de Ulong éléments dans lequel le ième élément correspond à l'orientation de la région de label i+1.

La valeur d'orientation est donnée en **degré [0,360]**.
Elle est calculée à partir des moments d'inertie:

```
orientation = 0.5*arctan(2*M11/(M20-M02)).
```

Si `M20=M02` alors c'est que la région présente une symétrie de rotation. Dans ce cas, la valeur d'orientation est égale à **360**.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné au tableau qui va contenir les valeurs d'orientation dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche l'orientation des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionorientation orientation b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionOrientation( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregionperimeter

Calcul du périmètre des régions.

Synopsis

pregionperimeter *attr* [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregionperimeter** crée une collection *col_out* contenant un tableau nommé *attr* de Ulong dans lequel le ième correspond au périmètre de la région de label i+1.

Le périmètre correspond au nombre de points internes d'une région qui sont sur la frontière interne de la région. Il est calculée. nombre de pixels de la frontière. A chaque creux interne dans une région, on ajoute un petit quart de pixel de périmètre. Dans l'exemple suivant, le périmètre est égal à 7= 6+ 4x.25.

```
 xx
xxxx
 xx
```

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de périmètre dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche le périmètre des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionperimeter perimeter b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionPerimeter( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregionrectangularity

Calcul de la rectangularité des régions.

Synopsis

pregionrectangularity *attr* [*-m mask*] [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregionrectangularity** crée une collection *col_out* contenant un tableau nommé *attr* de float dans lequel le ième correspond à la rectangularité de la région de label i+1.

La rectangularité est égale au rapport de la surface de la région sur la surface du rectangle exinscrit.

```
rectangularite = surface(rg_in[i])/surface(rectangleexinscrit(rg_in[i]))
```

Le rectangle exinscrit est calculé avec différentes orientations, et on conserve celui qui donne la rectangularité maximale, en considérant que c'est celui qui épouse le mieux la forme.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de rectangularité dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la rectangularité des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionrectangularity rectangularity b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
PRegioinRectangularity( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregionsphericity

Calcul de la sphéricité des régions.

Synopsis

pregionsphericity *attr* [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregionsphericity** crée une collection *col_out* contenant un tableau nommé *attr* de float dans lequel le ième correspond au degré de sphéricité de la région de label i+1.

La sphéricité est le rapport entre le rayon du cercle inscrit et le rayon du cercle circonscrit:

```
sphericite = rayon inscrit/rayon circonscrit.
```

Ce rapport vaut 1 pour un cercle.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de sphéricité dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la sphéricité des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionsphericity sphericity b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionSphericity( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Régis Clouard

pregionvariance

Calcul de la variance des régions.

Synopsis

pregionvariance *attr* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pregionvariance** crée une collection *col_out* contenant un tableau nommé *attr* de Double dans lequel le ième correspond à la variance calculée dans l'image *im_in* de la région numérotée i+1 dans *rg_in*.

La variance d'une région est calculée par la formule :

```
var = ((n*sigma2)-(sigma*sigma)) / (N*N)
```

où sigma est la somme des valeurs de niveau de gris de la région,
où sigma2 est la somme des carrés des valeurs de niveau de gris de la région et
où N est le nombre de pixels de la région.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de variance dans la collection.

Entrées

- *rg_in*: une carte de régions 2D.
- *im_in*: une image de niveaux de gris 2D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la variance des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionvariance variance b.pan tangram.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionVariance( const Reg2d &rg_in, const Img2duc &im_in,
Collection &cold, const std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregionvolume

Calcul du volume des régions.

Synopsis

pregionvolume *attr* [*-m mask*] [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregionvolume** crée une collection *col_out* contenant un tableau nommé *attr* de Ulong dans lequel le ième correspond au volume de la région de label i+1.

La valeur de volume est calculée en nombre de pixels inclus dans la région et sur la frontière.

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de volume dans la collection.

Entrées

- *rg_in*: une carte de régions 3D.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche le volume des régions obtenues par une simple binarisation de l'image tangram3d.pan:

```
pbinarization 100 1e30 tangram3d.pan a.pan
plabeling 8 a.pan b.pan
pregionvolume volume b.pan tangram3d.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionVolume( const Reg3d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregionwidth

Calcul de la largeur des régions.

Synopsis

pregionwidth *attr* [*-m mask*] [*rg_in*|-] [*col_out*|-]

Description

L'opérateur **pregionwidth** crée une collection *col_out* contenant un tableau nommé *attr* de Ulong dans lequel le ième correspond à la largeur de la région de label i+1.

La valeur de largeur est calculée en nombre de pixels

Paramètres

- *attr* est une chaîne de caractères qui définit le nom donné du tableau qui va contenir les valeurs de largeur dans la collection.

Entrées

- *rg_in*: une cartes de régions.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Affiche la largeur des régions obtenues par une simple binarisation de l'image tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionwidth area b.pan c.pan
pcol2txt c.pan
```

Voir aussi

Caractérisation de région

Prototype C++

```
Errc PRegionWidth( const Reg2d &rg_in, Collection &cold, const
std::string &attr );
```

Auteur: Alexandre Duret-Lutz

pregistrationPDE

Estimation du champ de déplacement entre deux images.

Synopsis

pregistrationPDE *smoothness precision* [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

Cet opérateur permet d'estimer un champ 2D de déplacement U entre deux images I1 et I2. Il se base sur la minimisation du critère :

```
E(U) = intégrale(|I1(p)-I2(p+U)| + smoothness*Laplac(U)).
```

Cette minimisation est effectuée par une succession de descente EDP à des échelles d'images différentes.

Le champ calculé correspond au déplacement de im_in1 vers im_in2.

Paramètres

Les paramètres de l'opérateur contrôlent l'estimation du champ de vecteurs déplacement :

- *smoothness* définit la régularité du champ calculé. Une valeur de 0 correspondant à aucune régularité particulière, 0.1 correspond à une régularité moyenne, et 0.9, une très forte régularité (champ quasi-constant). Si le déplacement entre deux images est connu pour être rigide (translation), une forte régularité est recommandée. Dans le cas plus génégal d'un déplacement non rigide, une régularité minimum est conseillée.
- *precision* définit le facteur de précision du calcul. Une précision élevée entraine des calculs plus long (le seuil de décision de la convergence pour chaque échelle sera plus fin). Une valeur de 0.9 définit déjà une bonne précision de calcul.

Entrées

- *im_in1*: une image 2D (Img2d,Imc2d ou Imx2d).
- *im_in1*: une image 2D (Img2d,Imc2d ou Imx2d).

Sorties

- *im_out* est une image multispectrale 2D de Float (Imx2dsf) comprenant deux bandes correspondantes aux composantes (vx,vy) des vecteurs déplacements estimés.

Résultat

Retourne SUCCESS ou FAILURE.

Exemple

Produit une translation horizontale de l’image "tangram.pan" puis affiche les vecteurs de déplacements

```
translation 0 10 tangram.pan tangram1.pan
pregistrationPDE 0.1 0.9 tangram.pan tangram1.pan a.pan
pplotquiver 800 800 10 -20 a.pan| visu
```

Voir aussi

Mouvement

Prototype C++

```
Errc PRegistrationPDE( const Imx2d &ims1, const Imx2d &ims2, Img2duc
&imd, Float smoothness, Float precision );
```

Avertissement

Auteur: D. Tschumperlé

pprelabelingfromarray

Relabelisation d'une carte de régions à partir des valeurs d'un vecteur d'etiquettes.

Synopsis

pprelabelingfromarray *attr* [*col_in*|-] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **prelabelingfromarray** permet de relabeliser la carte de régions *rg_in* à partir des valeurs du tableau (de `Ulong`) *attr* donnée dans *col_in*.

La région de label *x > 0* prend le nouveau label *attr[x-1]*. Si le tableau *attr* comporte moins de valeurs qu'il n'existe de régions dans *rg_in* alors les régions suplémentaires sont effacées. La région de label=0 reste à 0.

Entrées

- *col_in*: une collection.
- *rg_in*: une carte de régions

Sorties

- *rg_out*: une carte de régions.

Paramètres

- *attr* donne le nom du tableau à considérer dans le fichier *col_in*.

Résultat

Retourne le nombre de régions de la carte de sortie.

Exemples

Relabelise les régions à partir des valeurs du vecteur foo de la collection col.pan:

```
prelabelingfromarray foo col.pan rin.pn rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PPrelabelingFromArray( const std::string &attr, const
Collection &in, const Reg2d &reg_in, Reg2d &reg_out );
```

Auteur: Alexandre Duret-Lutz

prelabelingwithgraph

Renumérotation des régions d'une carte et des sommets du graphe associé.

Synopsis

prelabelingwithgraph [*rg_in*|-] [*gr_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

L'opérateur **prelabelingwithgraph** procède à une renumérotation des régions de *rg_in* et des sommets du graphe associé. Chacune des régions de *rg_in* se voit affecté d'un nouveau numéro dans *rg_out*, tel que tous les numéros entre 1 et n soient utilisés. Les numéros de sommets correspondants sont renumérotés de la même façon dans *gr_out* en gardant la correspondance numéro de sommet = numéro de région.

Remarque: Pour renuméroter une carte de régions sans le graphe de voisinage associé, il faut utiliser ***plabeling***.

Entrées

- *rg_in*: une carte de régions
- *gr_in*: un graphe

Sorties

- *rg_in*: une carte de régions
- *gr_out*: un graphe

Résultat

Retourne le nombre de régions de la carte de sortie.

Exemples

Relabelise les régions de la carte rin.pan et met à jour les arcs entre les noeuds du graphe g.pan:

```
relabelingwithgraph rin.pan g.pan rut.pn
```

Voir aussi

Région

Prototype C++

```
Errc PRelabelingWithGraph( const Reg2d &rg_in, Graph2d &gr_in, Reg2d
&rg_out, Graph2d &gr_out );
```

Auteur: Régis Clouard

premoveslice

Suppression d'un plan 2D dans une image 3D.

Synopsis

premoveslice *direction* [*-m mask*] [*im_in*|-] [*im_out1*|-] [*im_out2*|-]

Description

L'opérateur **premoveslice** supprime une image 2D de la fin ou du début d'une image 3D. L'image 2D est supprimée du début de l'image 3D si le paramètre *direction* est négatif ou de la fin s'il est positif et est retournee dans *im_out2*. La nouvelle image 3D *im_out1* a 1 plan de moins que l'image 3D d'entrée *im_in1*.

La dernière image 3D peut être castée en image 2D par l'opérateur pim3d22d.

L'image de sortie *im_out* est du même type que les deux images d'entree.

Pour le scartes de régions, il peut être judicieux de relabeliser les régions avec plabeling.

Paramètres

- *direction* spécifie si l'image doit être supprimée du début si *direction* < 0 ou de la fin si *direction* > 0 de l'image 3D.

Entrées

- *im_in*: une image 3D ou une carte de région 3D.

Sorties

- *im_out1*: une image 3D ou une carte de région 3D du même type que l'image d'entrée.
- *im_out2*: une image 2D ou une carte de région 2D.

Résultat

Retourne SUCCESS ou FAILURE au cas où l'entrée n'est pas de bon type ou l'image 3D n'a qu'un plan.

Exemples

Supprime et récupère dans l'image 2D a2d.pan le dernier plan de l'image 3D a3d.pan:

```
premoveslice 1 a3d.pan b3d.pan a2d.pan
```

Voir aussi

Utilitaire, pgetslice, paddslice, pim3d22d

Prototype C++

```
Errc PRemoveSlice( const Imx3d &im_in, const Imx2d &im_out1, Imx3d
&im_out2, int direction );
```

Auteur: Régis Clouard

prescale

Augmentation ou réduction de la taille d'une image, d'une carte de région ou d'un graphe.

Synopsis

prescale *zoomx zoomy zoomyz* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prescale** permet l'agrandissement ou la réduction de la taille d'une image d'un facteur *zoomx* selon l'axe x, *zoomy* selon l'axe y et *zoomz* selon l'axe z (pour les images 3D). L'image est agrandie selon un axe si le facteur de zoom est > 1 et réduite si le facteur de zoom est >0 et <1.

Cette version utilise l'interpolation au plus proche voisin. La réduction d'une image consiste en un sous-échantillonage des pixels de l'image d'entree et l'agrandissement consiste en une réplication des pixels :

```
im_out[z][y][x]=im_in[z/zoomz][y/zoomy][x/zoomx];
```

L'interpolation au plus proche voisin est la version la plus simple et la plus rapide des algorithmes de retaille d'image. Par contre, il provoque des effets de pavage sur l'image.

De meilleurs résultats peuvent être obtenus avec les opérateurs plinearrescale ou pbicubicrescale.

Paramètres

- *zoomx*, *zoomy*, *zoomz* sont des réels positifs correspondant aux facteurs de retaille.
  Si les zooms sont > 1 alors il s'agit d'un agrandissement, s'ils sont < 1 il s'agit d'une réduction.
  zoomz est ignoré pour le cas des images 2D mais doit être donné.

Entrées

- *im_in*: une image, une carte de régions ou un graphe.

Sorties

- *im_out*: une image de même type que l'objet d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Agrandissement de l’image d’un facteur 2 :

```
prescale 2 2 0 tangram.pan a.pan
```

Réduction de l’image d’un facteur 2 :

```
prescale 0.5 0.5 0 tangram.pan a.pan
```

Voir aussi

Transformation, pbilinearrescale, pbicubicrescale,

Prototype C++

```
Errc Prescale( const Img2duc &im_in, Img2duc &im_out, const float
zoomy, const float zoomx );
```

Auteur: Régis Clouard

presize

Ajustement de la taille d'une image en fonction d'une taille souhaitée.

Synopsis

presize *largeur hauteur profondeur* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **presize** permet de réduire ou d'augmenter la taille de l'image *im_in* par réduction ou interpolation linéaire en X,Y ou Z. L'image est réduite ou agrandie de telle façon qu'elle ait la taille (hauteur x largeur) en fin de réduction ou de l'interpolation.

Parce que la réduction peut provoquer des pertes d'informations telles que des points isolés, des lignes ... et l'aggrandissement des effets de pavés, il est génégalement nécessaire d'accompagner cet opérateur de lissage avant pour la réduction ou aprés pour l'agrandissement.

Paramètres

- *profondeur*,*hauteur*, *largeur* sont des valeurs entières. La profondeur doit être donnée pour une image 2D, mais elle est ignorée.

Entrées

- *im_in*: une image ou une carte de régions.

Sorties

- *im_out*: une image de même type que l'image d'entrée ou une carte de régions.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Retaille l'image tangram.pan pour l'agrandir à 512x256:

```
pmeanfiltering 1 tangram.pan a.pan
presize 512 256 0 a.pan b.pan
```

Voir aussi

Transformation

Prototype C++

```
Errc PResize( const Img2duc &im_in, Img2duc &im_out, int largeur,
int hauteur );
```

Auteur: Régis Clouard

prg2gr

Création d'un graphe d'un voisinage à partir d'une carte de régions.

Synopsis

prg2gr [*-m mask*] [*rg_in*|-] [*gr_out*|-]

Description

L'opérateur **prg2gr** crée le graphe d'adjacence des régions voisines dans *gr_out*, à partir de la carte de régions d'entrée *rg_in*.

Deux régions connexes dans *reg_in* vont correspondre à deux sommets reliés dans le graphe de sortie. Le sommet est positionné au centre de gravité de la région. *Il se peut alors qu'il ne soit pas sur la région proprement dite.*

Chaque arc est pondéré avec la valeur par défaut 1.0.

Remarque: Le graphe *gr_out* possède 1 sommet de plus que le nombre de régions de *rg_in* puisque le sommet 0 n'existe pas dans la représentation mais est présent dans la liste des sommets du graphe.

Entrées

- *rg_in*: une carte de régions.

Sorties

- *gr_out*: un graphe.

Résultat

Retourne SUCCESSS ou FAILURE.

Exemples

Construit le graphe g.pan à partir de la carte de régions r.pan

```
prg2gr r.pan g.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PRg2Gr( const Reg2d &rg_in, Graph &gr_out );
```

Auteur: François Angot

prg2im

Récupération de l'image d'étiquettes d'une carte de régions.

Synopsis

prg2im [*-m mask*] [*rg_in*|-] [*im_out*|-]

Description

prg2im récupère la carte d'étiquettes correspondant à la carte de régions *rg_in*. Les valeurs des étiquettes de l'image de sortie sont les mêmes que les valeurs de labels de l'image d'entrée. Par exemple, le label 10 devient le niveaux de gris 10.

L'image de sortie est donc une image de type signed long.

Entrées

- *rg_in*: une carte de régions.

Sorties

- *im_out*: une image de type Slong.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit l'image de labels out.pan à partir de la carte de région obtenue par seuillage de l'image tangran.pan.

```
pthresholding 100 1e30 tangram.pan a.pan
plabeling 8 .pan b.pan
prg2im b.pan out.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PRg2Im( const Reg2 &rg_in, Img2dsl &im_out );
```

Auteur: Régis Clouard

prg2imc

Conversion d'une carte de régions en image couleur équivalente.

Synopsis

prg2imc [*-m mask*] [*rg_in*|-] [*im_out*|-]

Description

prg2imc construit une image couleur qui visualise les régions de la carte de régions *rg_in*. Les couleurs choisies pour visualiser les régions sont des fausses couleurs choisies arbitrairement.

Entrées

- *rg_in*: une carte de régions.

Sorties

- *im_out*: une image couleur (Imc2duc).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit une image qui visualise les régions extraites par seuillage de l'image tangran.pan.

```
pthresholding 100 1e30 tangram.pan a.pan
plabeling 8 .pan b.pan
prg2imc b.pan out.pan
```

Voir aussi

Coercition

Prototype C++

```
Errc PRg2Imc( const Reg2 &rg_in, Img2dsl &im_out );
```

Auteur: Régis Clouard

prgb2ast

Changement d'espace couleur de RGB vers AST.

Synopsis

prgb2ast [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2ast** construit une nouvelle image couleur *im_out* par conversion de l'image couleur *im_in* au format RGB en image couleur au format AST.

L'espace couleur de Chassery est un modèle simplifié basé sur les composantes:

$$A = \frac{\log(R) + \log(V) + \log(B)}{3}$$
$$C_1 = \sqrt{3}/2 * (\log(R) - \log(G))$$
$$C_2 = \log(B) - 1/2*(\log(R)+\log(G))$$

A est une composante achromatique A et C1 et C2 deux composantes chromatiques. A partir de ce modèle on peut ensuite calculer la saturation et la teinte données par:

$$s = \sqrt{C1^2 + C2^2}$$
$$t = \arccos(C1/s)$$

Entrées

- *im_in*: une image couleur RGB.

Sorties

- *im_out*: une image couleur AST.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
prgb2ast parrot.pan a.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2AST( const Imc2duc &im_in, Imc2duc &im_out );
```

Reference

Reference: J.M. Chassery, "An iterative segmentation method based on a contextual color and shape criterion", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 6, No. 6, pp 794-800, 1984.

Auteur: Olivier Lezoray

prgb2cmyk

Changement d'espace couleur de RGB vers CMYK (Cyan-Magenta-Yellow-Key).

Synopsis

prgb2cmyk [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2cmyk** permet de changer l'espace Rouge-vert-bleu, vers l'espace Cyan-Magenta-Yellow-Key.

Le modèle de couleur CMYK est un modèle soustractif, utilisé en imprimerie. CMYK se rapporte aux quatre types d'encre utilisés pour imprimer: cyan, magenta, yellow, et key (noir).

La conversion procède selon l'algorithme suivant:

```
if (R=0 and G=0 and B==0) then C=0; M=0; Y=0; K = 255;
else
  x = 1 - (R/255);
  y = 1 - (G/255);
  z = 1 - (B/255);

  min = MIN(x, MIN(y,z));
  C = (x - min) / (1 - min) *255;
  M = (y - min) / (1 - min) *255;
  Y = (z - min) / (1 - min) *255;
  K = min * 255;
```

Entrées

- *im_in*: une image couleur RGB.

Sorties

- *im_out*: une image multispectrale à 4 bandes.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit parrot.pan de rgb en cmyk, puis récupération de la bande Cyan.

```
prgb2cmyk parrot.pan a.pan
pgetband 0 a.pan cyan.pan
```

Voir aussi

Color, pcmyk2rgb

Prototype C++

```
Errc PRgb2Cmyk( const Imc2duc &im_in, Imc2duc &im_out );
```

Auteur: Régis Clouard

prgb2gray

Changement d'espace couleur de RGB vers niveaux de gris.

Synopsis

prgb2gray *red green blue* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2gray** permet de construire une image de niveaux de gris par combinaison des valeurs de couleur.

L'algorithme est le suivant:

```
pixel(im_out) = red*pixel.X(im_in)+green*pixel.Y(im_in)+blue*pixel.Z(im_in)/(red+green+blue);
```

L'image de sortie est une image de niveaux de gris de même type que l'image couleur d'entrée.

Exemple : Méthode standard NTSC

```
red=0.299; green=0.587; blue=0.114;
```

Paramètres

- *red* spécifie le ratio de la composante rouge.
- *green* spécifie le ratio de la composante verte.
- *blue* spécifie le ratio de la composante bleu.

Entrées

- *im_in*: une image couleur RGB.

Sorties

- *im_out*: une image de niveaux de gris.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit parrot.pan den niveaux de gris en utilisant le standard de conversion NTSC:

```
prgb2gray 0.299 0.587 0.114 parrot.pan a.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2Gray( const Imc2duc &im_in, Img2dsf &im_out, float red,
float green, float blue );
```

Auteur: Régis Clouard

prgb2hsi

Changement d'espace couleur de RGB vers HSI.

Synopsis

prgb2hsi [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2hsi** permet de changer l'espace couleur (Rouge, Vert, Bleu) vers l'espace HSI (Teinte, Saturation, Intensité).
La teinte (Hue) est la qualité de couleur correspondant à sa position dans le spectre: rouge, orange, jaune, vert, cyan, bleu, magenta. Elle s'exprime en degré [0,360].
La saturation est l'intensité d'une couleur. Elle s'exprime par une valeur d'intensité de pourcentage de l'intervalle [0..100]. A 0% de saturation, une couleur apparaît blanche, à 100% de saturation, une couleur atteint son degré de plus intense.
L'intensité (Intensity) est la quantité de blanc et de noir contenue dans une couleur. Elle s'exprime par une valeur d'intensité de l'intervalle [0,255].

L'image de sortie est par conséquent de type float.

Le principe de transformation d'une composant RGB en HSI :

```
          [         ((R-V)+(R-B))           ]
H = acos  [ ------------------------------  ]
          [ 2*sqrt((R-V)*(R-V) + (R-B)(V-B)) ]
S = 1 - 3*MIN(R,V,B)
        ------------
        (R + V + B)
L = (R + V + B)/3
```

Ainsi, les couleurs primaires suivantes ont pour valeur <H,S,L>:

Rouge: < 0 , 1 , 85>
Vert: <120 , 1 , 85>
Bleu: <240, 1 , 85> Noir: <90, 1 , 0>

Entrées

- *im_in*: une image couleur RGB.

Sorties

- *im_out*: une image couleur HSI.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit parrot.pan de rgb en hsi et réciproquement.

```
prgb2hsi parrot.pan a.pan
phsitogb a.pan b.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2HSI( const Imc2duc &im_in, Imc2dsf &im_out );
```

Auteur: Olivier Lezoray

prgb2hsl

Changement d'espace couleur de RGB vers HSL.

Synopsis

prgb2hsl [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2hsl** permet de changer l'espace couleur (Rouge, Vert, Bleu) vers l'espace HSL (Teinte, Saturation, Luminance).

La teinte (Hue) est la qualité de couleur correspondant à sa position dans le spectre: rouge, orange, jaune, vert, cyan, bleu, magenta. Elle s'exprime en degré [0,360].

La saturation est l'intensité d'une couleur. Elle s'exprime par une valeur d'intensité de pourcentage de l'intervalle [0..100]. A 0% de saturation, une couleur apparaît blanche, à 100% de saturation, une couleur atteint son degré de plus intense.

La luminosité (Lightness) est la quantité de blanc et de noir contenue dans une couleur. Elle s'exprime par une valeur d'intensité de l'intervalle [0,255].

L'image de sortie est par conséquent de type float.

Le principe de transformation d'une composant RGB en HSL :

```
  Soit max = MAX(R,G,B) et min = MIN(R,G,B)

   |- 0                            si max = min
   |         (G-B)
H= |- (60 *  ----- + 360) mod 360 si max = R
   |       (max-min)
   |         (B-R)
   |- (60 *  ----- + 120) + 210   si max = V
   |       (max-min)
   |         (R-G)
   |- (60 *  ----- + 240)         si max = B
   |        (max-min)


     (max+min)
L=   --------
         2

    |- 0                 si max = min
    |
    |       max-min
    |- 100 *-------      si l<=1/2
S = |       max+min
```

```
|
|         max-min
|- 100 * ----------- si l>1/2
|        2-(max+min)
```

Entrées

- *im_in*: une image couleur RGB.

Sorties

- *im_out*: une image couleur HSL.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit parrot.pan de rgb en hsl et réciproquement.

```
prgb2hsl parrot.pan a.pan
phsltorgb a.pan b.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2HSL( const Imc2duc &im_in, Imc2dsf &im_out );
```

Auteur: Régis Clouard

prgb2hsv

Changement d'espace couleur de RGB vers HSV.

Synopsis

prgb2hsv [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2hsv** permet de changer l'espace couleur (Rouge, Vert, Bleu) vers l'espace HSV (Teinte, Saturation, Valeur).

La teinte (Hue) est la qualité de couleur correspondant à sa position dans le spectre: rouge, orange, jaune, vert, cyan, bleu, magenta. Elle s'exprime en degré [0,360].

La saturation est l'intensité d'une couleur. Elle s'exprime par une valeur d'intensité de pourcentage de l'intervalle [0..100]. A 0% de saturation, une couleur apparaît blanche, à 100% de saturation, une couleur atteint son degré de plus intense.

La valeur est la plus forte composante couleur. Elle s'exprime par une valeur d'intensité de l'intervalle [0,255].

L'image de sortie est par conséquent de type float.

Le principe de transformation d'une composant RGB en HSV :

```
  Soit max = MAX(R,G,B) et min = MIN(R,G,B)

    |- 0                              if max = min
    |          (G-B)
H = |- (60 * -------- + 360) mod 360 if max = R
    |        (max-min)
    |          (B-R)
    |- (60 * -------- + 120) + 210   if max = V
    |        (max-min)
    |          (R-G)
    |- (60 * -------- + 240)         if max = B
           (max-min)

V =  max

    |- 0               si max = 0
S = |
    |         max-min
    |- 100 * ---------  si max > 0
               max
```

Entrées

- *im_in*: une image couleur RGB.

Sorties

- *im_out*: une image couleur HSV.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit parrot.pan de rgb en hsv et réciproquement.

```
prgb2hsv parrot.pan a.pan
phsvtorgb a.pan b.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2HSV( const Imc2duc &im_in, Imc2dsf &im_out );
```

Auteur: Régis Clouard

prgb2i1i2i3

Changement d'espace couleur RGB vers (i1,i2,i3).

Synopsis

prgb2i1i2i3 [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2i1i2i3** permet de passer de l'espce couleur à l'espace i1,i2,i3.

Entrées

- *im_in*: une image couleur au format RGB.

Sorties

- *im_out*: une image couleur de float au format i1i2i3.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
prgb2i1i2i3 parrot.pan a.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2I1I2I3( const Imc2duc &Ims, Imc2dsf & Imd );
```

Auteur: Olivier Lezoray

prgb2pca

Calcul des composantes principales d'une image couleur.

Synopsis

prgb2pca [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2pca** permet de changer d'espace couleur en utilisant la transformée de Karhunen-Loeve qui donne les composantes principales d'une image.

Entrées

- *im_in*: les images couleur RGB.

Sorties

- *im_out*: une image couleur.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
prgb2pca parrot.pan a.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2PCA( const Imc2duc &im_in, Imc2duc &im_out );
```

Auteur: Olivier Lezoray

prgb2rngnbn

Changement d'espace couleur de RGB vers RGB normalisé.

Synopsis

prgb2rngnbn [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgbrngnbn** permet de passer de l'espace couleur RGB à l'espace couleur RGB normalisé.

Chaque valeur de couleur est divisée par la somme des valeurs des trois couleurs.

L'opérateur consiste pour chaque pixel:

```
rouge(im_out)=rouge(im_in)/(rouge(im_in)+vert(im_in)+bleu(im_in))
vert(im_out)=vert(im_in)/(rouge(im_in)+vert(im_in)+bleu(im_in))
bleu(im_out)=bleu(im_in)/(rouge(im_in)+vert(im_in)+bleu(im_in))
```

Entrées

- *im_in*: les images couleur RGB.

Sorties

- *im_out*: une image couleur RNGNBN.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
prgb2rngnbn parrot.pan a.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2RNGNBN( const Imc2duc &im_in, Imc2dsf &im_out );
```

Auteur: Olivier Lezoray

prgb2wry

Changement d'espace couleur de RGB à (wb,rg,yb).

Synopsis

prgbwry [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2wry** de passer de l'espace couleur RVB à l'espace couleur (wb,rg,yb) défini par Swain & Ballard.

Entrées

- *im_in*: une image couleur au format RGB.

Sorties

- *im_out*: une image couleur de floats au format WRY.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
prgb2wry parrot.pan a.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2WRY( const Imc2duc &im_in, Imc2dsf &im_out );
```

Auteur: Olivier Lezoray

prgb2xyz

Changement d'espace couleur de RGB vers XYZ.

Synopsis

prgb2xyz *primaries* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2xyz** permet de changer l'espace RGB vers l'espace X,Y,Z. En XYZ, chaque valeur est représentée par un ensemble de valeurs positives entre 0..1.

L'algorithme de conversion utilise la matrice de transformation pour le cas *primaries* = 4 (illuminant C CIE):

```
|  0.607   0.174   0.200 |
|  0.299   0.587   0.114 |
|  0.000   0.066   1.116 | / VALMAX
```

où VALMAX est la valeur maximale du type (ex: 255 pour les images d'octets).

Paramètres

- *primaries* est un entier de l'espace [0..6] qui définit le type de conversion :
  - 0-illuminant E
  - 1-illuminant primaries CIE-DIN
  - 2-illuminant A primaries macbeth colour chart
  - 3-illuminant A primaries CIE
  - 4-illuminant C primaries NTSC
  - 5-illuminant C primaries CIE
  - 6-illuminant D65

Entrées

- *im_in*: une image couleur RGB.

Sorties

- *im_out*: une image couleur XYZ.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit parrot.pan de rgb en xyz and réciproquement.

```
prgb2xyz 4 parrot.pan a.pan
pxyz2rgb 4 a.pan b.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2XYZ( const Imc2duc &im_in, Imc2dsf &im_out, int primaries
);
```

Auteur: Olivier Lezoray

prgb2ycbcr

Changement d'espace couleur de RVB vers YCbCr.

Synopsis

prgb2ycbcr [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2ycbcr** permet de passer de l'espace couleur RGB à l'espace couleur YCbCr.

Entrées

- *im_in*: les images couleur RGB.

Sorties

- *im_out*: une image couleur YCbCr.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
prgb2ycbcr parrot.pan a.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2YCBCR( const Imc2duc &im_in, Imc2dsf &im_out );
```

Auteur: Olivier Lezoray

prgb2ych1ch2

Changement d'espace couleur de RGB vers YCh1Ch2.

Synopsis

prgb2ych1ch2 [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2ych1ch2** permet de passer de l'espace couleur RVB à l'espace couleur YCh1Ch2 (Système de Carron).

Entrées

- *im_in*: les images couleur RGB.

Sorties

- *im_out*: une image couleur YCh1Ch2.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
prgb2ych1ch2 parrot.pan a.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2YCH1CH2( const Imc2duc &im_in, Imc2dsf &im_out );
```

Auteur: Olivier Lezoray

prgb2yiq

Changement d'espace couleur de RVB vers YIQ.

Synopsis

prgb2yiq [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2yiq** permet de passer de l'espace couleur RGB (Rouge, Vert, Bleu) à l'espace couleur YIQ(standard de télévision). L'espace couleur YIQ est le système de couleurs primaires dopt pr NTSC comm standrd de couleur de télévision. Le solide des couleurs YIQ est fait pr transformation linéaire du cube RGB.

Le but de cet espace est d'exploiter certaines caractéristiques de l'oeil humain pour maximiser l'utilisation d'une langueur de bande fixe. L'oeil humain est en effet plus sensible aux changements de luminance qu'aux changements de la teinte ou de la saturation.

La conversion utilise la transformation linéaire suivante :

```
Y =     0.299Red    + 0.587Green        + 0.114Blue
I =     0.595716Red - 0.274453Green     - 0.321263Blue
Q =     0.211456Red - 0.522591Green     + 0.311135Blue
```

Entrées

- *im_in*: les images couleur RGB.

Sorties

- *im_out*: une image couleur YIQ.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
prgb2yiq a.pan b.pan
```

Voir aussi

Color

Prototype C++

```
Errc PRGB2YIQ( const Imc2duc &im_in, imc2dsf &im_out );
```

Auteur: Olivier Lezoray

prgb2yuv

Changement d'espace couleur de RVB vers YUV.

Synopsis

prgb2yuv [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **prgb2yuv** permet de passer de l'espace couleur RGB (Rouge, Vert, Bleu) à l'espace couleur Yuv (standard de télévision). L'espace couleur Yuv est l'espace couleur adopté pour le format télévision Pal.

La conversion de couleur est un opération linéaire :

```
|  0.299   0.587   0.114 |
| -0.147  -0.289   0.437 |
|  0.615  -0.515  -0.10 |
```

Entrées

- *im_in*: les images couleur RGB.

Sorties

- *im_out*: une image couleur Yuv.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
prgb2yuv parrot.pan a.pan
```

Voir aussi

pyuv2rgb, Color

Prototype C++

```
Errc PRGB2YUV( const Imc2duc &im_in, Imc2dsf &im_out );
```

Auteur: Olivier Lezoray

proberts

Module du gradient de Roberts.

Synopsis

proberts [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **proberts** permet d'approximer le calcul de l'amplitude du gradient de l'image *im_out*.

L'algorithme consiste à convoluer l'image par le masque de Roberts:

```
|+0 -1|
|+1 +0|
```

En fait ici, le masque est orienté successivement dans les deux directions: 0, 90 degrés et c'est la valeur maximale qui est choisie comme amplitude.

L'image de sortie *im_out* est de même type que l'image d'entrée *im_in*.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image du même type que l'image im_in.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Détection de contours de l'image tangram.pan:

```
proberts tangram.pan b.pan
pbinarization 15 1e30 b.pan out.pan
```

Voir aussi

Détection de contours

Prototype C++

```
Errc PRoberts( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

protation

Construction de la rotation du contenu d'une image selon un axe.

Synopsis

protation *axe angle* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **protation** contruit une image *im_out* qui est la rotation d'un angle *angle* autour d'un l'*axe* du contenu de l'image d'entrée *im_in*.

L'image *im_out* conserve les mêmes dimensions que l'image *im_in*.

Les nouveaux pixels introduits dans l'image sont de valeur = 0.

Remarque: une rotation de 4 fois 90 degrés donne une image légèrement différente de l'image de départ à cause des approximations des calculs en flottant.

Paramètres

- Le paramètre *angle* est un réel mesurée en degr´ et la valeur peut être négative.
- *axe* est identifié par un entier:
  - 0: rotation autour de l'axe z.
  - 1: rotation autour de l'axe y.
  - 2: rotation autour de l'axe x.

  Pour les images 2D, seule la valeur 0 est prise en compte.

Entrées

- *im_in*: une image ou une carte de régions.

Sorties

- *im_out*: une image de même type que l'image d'entrée ou une carte de régions

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Tourne l'image tangram.pan de 90 degree.

```
protation 0 90 tangram.pan a.pan
```

Voir aussi

Transformation

Prototype C++

```
Errc PRotation( const Img2duc &im_in; Img2duc &im_out; int axe,
float angle );
```

Auteur: Régis Clouard

pround

Arrondi d'une image de réels ou d'un graphe.

Synopsis

pround *mode* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pround** construit l'arrondi d'une image de réels. Chaque pixel de l'image de sortie *im_out* est construit avec la valeur arrondie du pixel correspondant dans l'image d'entrée *im_in*.

```
pixel(im_out)=arrondi(pixel(im_in))
```

Le type d'arrondissement dépend du paramètre *mode*.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

L'image de sortie est du même type que l'image d'entrée.

Pour les graphes, le graphe de sortie est construit avec la valeur arrondie des valeurs de noeuds.

Paramètres

- *mode* indique le type d'arrondissement:
  - 0: le plus proche entier (1.1=1; 1.9=2; -8.8=-9; -8.1=-8).
  - 1: l'arrondi par défaut (1.1=1, 1.9=1; -8.8=-9; -8.1=-9);
  - 2: l'arrondi par excès (1.1=2; 1.9=2; -8.8=-8; -8.1=-8);

Entrées

- *im_in*: une image de Floats ou un graphe.

Sorties

- *im_out*: une image de Floats ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
plog a.pan b.panx
pround 0 b.pan c.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PRound( const Img2dsf &im_in, Img2dsf &im_out, int mode );
```

Auteur: Régis Clouard

pscrolling

Construction de l'enroulé d'une image.

Synopsis

pscrolling *direction decalage* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pscrolling** permet de construire une nouvelle image *im_out* en translatant le contenu de l'image *im_in* de *decalage* pixels dans la *direction* indiquée.

Les pixels poussés hors de l'image se retrouvent de l'autre côté de l'image de sortie.

Paramètres

- *direction* donne la direction de translation:
  - 0 = translation en x.
  - 1 = translation en y.
  - 2 = translation en z.
- *decalage* est une valeur entière positive ou négative selon le sens du décalage souhaité.

Entrées

- *im_in*: une image ou une carte de régions.

Sorties

- *im_out*: une image de même type que l'image d'entrée ou une carte de régions.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Translate le contenu de l'image tangram.pan de 50 pixels à gauche:

```
pscrolling 0 50 tangram.pan a.pan
```

Voir aussi

ptranslation, Transformation

Prototype C++

```
Errc PScrolling( const Img3duc &im_in,Img3duc &im_out, Uchar
direction, Long decalage );
```

Auteur: Régis Clouard

psedesign

Génération d'un élément structurant prédéfini.

Synopsis

psedesign *num_se halfsize* [*im_out*|-]

Description

L'opérateur **psedesign** permet de générer un élément structurant en tant qu'image Pandore.

L'élément structurant dépend du type *num_se* et de sa demi-taille par *halfsize*.

Il est possible de construire un élément structurant de forme quelconque directement à partir de sa spécification dans un texte texte (voir ptxt2pan).

Ces éléments structurants peuvent ensuite être utilisés par les opérateurs `erosionse` et `dilationse`.

Paramètres

- *num_se* spécifie le type de l'élément structurant :

  En 2D:

  - 0: losange (4-connexité)
  - 1: carré (8-connexité).
  - 2: cercle
  - 3: ligne horizontal
  - 4: ligne diagonale de 135 degrees (\)
  - 5: ligne verticale
  - 6: ligne diagonale de 45 degrees (/)
  - 7: croix (+)
  - 8: X

  En 3D:

  - 10: bipyramide (6-connexité)
  - 11: cube (26-connexité)
  - 12: sphère
  - 13: ligne horizontale sur l'axe x
  - 14: ligne horizontale sur l'axe y
  - 15: ligne horizontale sur l'axe z
  - 16: ligne diagonale sur le repère x-y (\)

    - 17: ligne diagonale sur le repère x-z (\)
    - 18: ligne diagonale sur le repère y-z (\)
    - 19: ligne diagonale sur le repère x-y (/)
    - 20: ligne diagonale sur le repère x-z (/)
    - 21: ligne diagonale sur le repère y-z (/)
    - 22: croix en 3d
- *size* donne la demi-taille de l'élément structurant. Par exemple, une demi-taille de 1 pour un carré donne un structurant de taille 3x3.

Sorties

- *im_out* : une image 2D ou 3D d'octets.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Chapeau haut-de-forme noir géodésique avec un élément structurant carré de taille 17.

```
psedesign 1 8 es.pan
pinverse tangram.pan i0.pan
psedilation 1 i0.pan i1.pan
pdilationreconstruction 8 i1.pan i0.pan i2.pan
pdif i2.pan out.pan
```

Voir aussi

Morphologie, psedilation, pseerosion, ptxt2pan

Prototype C++

```
Img2duc* PSEDesign( int numse, int halfsize );
```

Auteur: Régis Clouard

psedilation

Dilatation des points de fort contraste d'une image à partir d'un élément structurant donné.

Synopsis

psedilation *size* [*-m mask*] [*im_se*|-] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **psedilation** permet de dilater les points de plus fort contraste de l'image *im_in* à partir d'un élément structurant donné dans l'image *im_se*.

La dilatation correspond à l'opération:

```
dilatation(x,y)= Max(voisins selon l'élément structurant de x,y).
```

Pour une image binaire cela revient à dilater les régions blanches.

Pour les cartes de régions, la dilatation ajoute des pixels de label=0 aux points de dilatation.

Pour les images couleur, c'est l'ordre lexicographique qui est utilisé: d'abord en utilisant la bande X, en cas d'égalité en utilisant la bande Y puis la bande Z.

L'élément structurant est donné sous la forme d'une image Pandore de type Uchar de la taille de l'élément structurant. Cette image peut être construite à partir d'un fichier texte (voir ptxt2pan) ou à partir du générateur de masque (voir psedesign).

Paramètres

- *size* donne le nombre d'itération de la dilatation à opérer.

Entrées

- *im_in*: une image ou une carte de régions.
- *im_se*: une image d'octets.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Chapeau haut-de-forme noir géodésique avec un structurant carré de taille 17.

```
psedesign 1 8 es.pan
pinverse tangram.pan i0.pan
psedilation 1 es.pan i0.pan i1.pan
perosionreconstruction 8 i1.pan i0.pan i2.pan
pdif i0.pan i2.pan out.pan
```

Voir aussi

Morphologie, pdilation, pseerosion

Prototype C++

```
Errc PSEDilation( const Img2duc &im_in, const Img2duc &im_se,
Img2duc &im_out, int size );
```

Auteur: Régis Clouard

pseedplacement

Placement de germes de régions sur une grille régulière.

Synopsis

pseedplacement *dx dy dz* [*-m mask*] [*im_in*|-] [*reg_out*|-]

Description

L'opérateur créer une carte de régions de la même taille que l'image d'entrée *im_in* avec des germes (région de taille 1) positionnés sur une grille régulière.

Paramètres

- *dx*, *dy*, *dz* donne la périodicité des germes en colonne, ligne et profondeur.

Entrées

- *im_in* : une image qui n'est utilisée que pour récupérer la taille de l'image.

Sorties

- *reg_out* : une carte de régions avec les germes.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

On place des germes tous les 10 pixels en largeur et en hauteur, puis on fait une sursegmentation par superpixels :

```
./pseedplacement 10 10 0 examples/tangram.pan rgin.pan
./psuperpixelsegmentation 0 1 rgin.pan examples/tangram.pan rgout.pan geodesic-map.pan
pboundary 8 rgout.pan cont.pan
psuperimposition 3 cont.pan examples:tangram.pan result.pan
```

Voir aussi

Segmentation, psuperpixelsegmentation

Prototype C++

```
Errc PSeedPlacement2D( Reg2d &rgd, int dx, int dy );
```

Auteur : Pierre Buyssens

pseerosion

Erosion des points de fort contraste d'une image à partir d'un élément structurant donné.

Synopsis

pseerosion *size* [*-m mask*] [*im_se*|-] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **erosiones** permet d'éroder les points de plus fort contraste de l'image *im_in* à partir d'un élément structurant donné dans l'image *im_se*.

L'érosion correspond à l'opération:

```
erosion(x,y)= Min(voisins selon l'élément structurant de x,y).
```

Pour une image binaire cela revient à éroder les régions blanches.

Pour les cartes de régions, l'érosion ajoute des pixels de label=0 aux points d'érosion.

Pour les images couleur, c'est l'ordre lexicographique qui est utilisé: d'abord en utilisant la bande X, en cas d'égalité en utilisant la bande Y puis la bande Z.

L'élément structurant est donné sous la forme d'une image Pandore de type Uchar de la taille de l'élément structurant. Cette image peut être construite à partir d'un fichier texte (voir ptxt2pan) ou à partir du générateur de masque (voir psedesign).

Paramètres

- *size* donne le nombre d'itération de l'érosion à opérer.

Entrées

- *im_in*: une image ou une carte de régions.
- *im_se*: une image d'octets (image Uchar).

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Chapeau haut de forme blanc géodésique avec un structurant carré de taille 17.

```
psedesign 1 8 es.pan
pseerosion 1 es.pan tangram.pan i1.pan
pdilationreconstruction 8 i1.pan tangram.pan i2.pan
pdif tangram.pan i2.pan out.pan
```

Voir aussi

Morphologie psedilation, perosion

Prototype C++

```
Errc PSEErosion( const Img2duc &im_in, const Img2duc &im_se, Img2duc
&im_out, int size );
```

Auteur: Régis Clouard

psetband

Récupération une bande dans une image multispectrale.

Synopsis

pgetband *band* [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pgetband** permet de créer une nouvelle image de niveaux de gris *im_out* à partir d'une bande de l'image d'entrée *im_in*. Le type des pixels de l'image de sortie *im_out* est le même que celui de l'image d'entrée. *im_in*.

Paramètres

- *band* est un entier. Si sa valeur est supérieure ou inférieure au nombre de bandes de l'image d'entrée alors la bande la plus proche est utilisée (première ou dernière bande).

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image en niveaux de gris.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Modifie la bande 0 de l'image parrot.pan:

```
pgetband 0 parrot.pan a.pan
pinverse a.pan b.pan
psetband 0 b.pan parrot.pan c.pan
```

Voir aussi

Utilitaire, pgetband.

Prototype C++

```
Errc PSetBand( const Img2duc &ims, const Imx2duc &ims2, int band);
```

Auteur: Régis Clouard

psetborder

Affectation d'une valeur sur le bord d'une image.

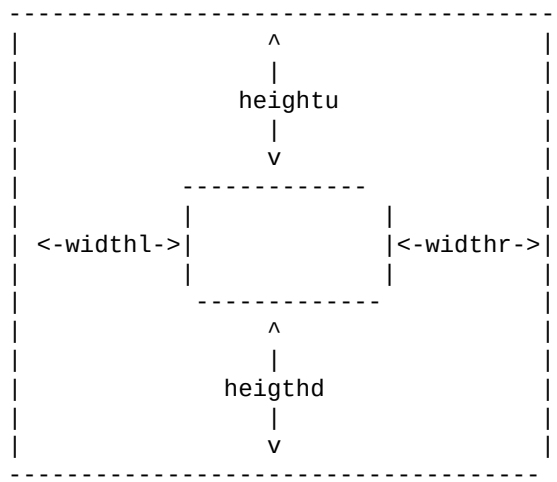
Synopsis

psetborder *widthl widthr heightu heightd depthf depthb val* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **psetborder** permet d'affecter la valeur *val* sur la bordure de l'image *im_in*. La bordure d'une image 3D à une taille de largeur *widthl* pixels à gauche et *widthr* pixels à droite, de hauteur *heightu* pixels en haut et *heightd* pixels en bas et de profondeur *depthf* pixels devant et *depthb* pixels derrière.

Pour une image 2D, les dimensions sont:



Pour une image couleur ou multispectrale, la bordure est modifée avec la même valeur *val* sur toutes les bandes.

Paramètres

- *depthf, depthb, heightu, heightd, widthl, widthr* donnent la taille en pixels de chacune des trois bordures en profondeur, en hauteur et en largeur.

  Dans le cas 2D, *depthf et depthb* ne sont pas utilisés mais doivent être donnés.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image de même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Sélection des objets d'une sous image de tangram.pan qui ne touchent pas le bord de l'image.

```
pextractsubimage  30 30 0 150 150 0 tangram.pan in.pan
psetcst 0 in.pan i1.pan
psetborder 1 1 1 1 1 1 255 i1.pan i2.pan
pdilationreconstruction 8 i2.pan in.pan i3.pan
pdif in.pan i3.pan out.pan
```

Voir aussi

Utilitaire

Prototype C++

```
Errc PSetBorder( const Img3duc &im_in, Img3duc &im_out, Long widthl,
Long widthr, Long heightu, Long heigthd, Long depthf, Long depthb,
Uchar val );
```

Auteur: Régis Clouard

psetcst

Affectation d'une valeur à une image, un graphe ou une carte de région.

Synopsis

psetcst *cst* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

psetcst construit l'image de sortie *im_out* en remplacant chaque valeur de l'image d'entrée par la valeur du paramètre *cst*.

Si *im_in* est une image, **psetcst** affecte chaque pixel avec la valeur *cst*.

Pour les images couleur ou multispectrale, **psetcst** est effectué séparément sur chaque bande.

Pour les cartes de régions, **psetcst**affecte chaque valeur de label. Il n'y a donc qu'une seule région en sortie.

Pour les graphes, **psetcst** affecte chaque valeur de noeud.

L'image de sortie est du même type que l'image d'entrée.

Paramètres

- *cst* est une valeur réelle. Si nécessaire, la constante est castée pour correspondre au type des valeurs de pixels de l'image d'entrée.

Entrées

- *im_in*: une image, un graphe ou une carte de régions

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Pour les cartes de régions, retourne la valeur de label maximale (ie. cst).

Exemples

Crée l'image a.pan avec les mêmes propriétés que l'image tangram.pan et affecte la valeur 10 à chaque pixel.

```
psetscst 10 tangram.pan a.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PSetCst( const Img2duc &im_in, Img2duc &im_out, Uchar cst );
```

Auteur: Régis Clouard

ppixelvalue

Affecte une valeur particulière à un pixel donné.

Synopsis

psetpixel *x y z value* [*im_in*|-] [*im_out*|-]

Description

psetpixel affecte la valeur *value* au pixel de coordonée *x*, *y*, *z*.

pour les images couleur et multispectrales, la valeur est affectée à toutes les bandes.

Paramètres

- *x*, *y*, *z* spécifie les coordonnées du pixel.

  dans le cas d'images 2D, *z* doit n'est pas utilisé, mais il doit être donné.

- *value* dépend du type de l'image d'entrée (float, integer, etc).

Entrées

- *im_in*: une image, une carte de régions ou un graphe.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne la valeur de pixel ou FAILURE

Exemples

Met la valeur 255 au pixel de coordonnées 10, 12 dans l'image tangram.pan :

```
psetpixel 10 12 0 255 examples/tangram.pan a.pan
```

Voir aussi

Utilitaire

Prototype C++

```
Errc PSetPixel( const Img3duc &im_in, Img3duc &im_out, Long z, Long
y, Long x, Uchar value );
```

Auteur: Régis Clouard

psetslice

Remplacement d'un plan d'une image 3D par une image 2D.

Synopsis

psetslice *slice* [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **psetslice** construit une image 3D *im_out* à partir de l'image 3D *im_in1* à laquelle a été remplacée le plan numéro *slice* par l'image 2D *im_in2*.

L'image résultat *im_out* est du même type que les deux images d'entrée.

Paramètres

- *slice* spécifie le numéro du plan à replacer. C'est un entier entre 0 et le nombre de plans moins 1.

Entrées

- *im_in1*: une image 3D ou une carte de régions 3D.
- *im_in2*: une image 2D ou une carte de régions 2D.

Sorties

- *im_out*: une image 3D ou une carte de régions 3D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Remplace le plan no. 10 (11e plan) de l'image a3d.pan avec l'image 2D a2d.pan:

```
psetslice 10 a3d.pan a2d.pan b3d.pan
```

Voir aussi

Utilitaire, pgetslice, premoveslice

Prototype C++

```
Errc PSetSlice( const Img3duc &im_in1, const Img2duc &im_in2,
Img3duc &im_out, long slice );
```

Auteur: Régis Clouard

psetstatus

Affectation d'une valeur au statut courant.

Synopsis

psetstatus *valeur*

Description

L'opérateur **psetstatus** affecte la valeur qui sera utilisée par la prochaine utilisation de l'opérateur 'pstatus'.

Cette commande est particulièrement utile pour construire de nouveaux opérateurs Pandore à partir de scripts Shell ou MsDOS qui retournent un résultat comme tout opérateur Pandore.

Le script Shell suivant devient un opérateur Pandore qui permet de tester si une image existe:

```
#!/bin/sh
if [ -f $1 ]
then
   psetstatus SUCCESS
else
   psetstatus FAILURE
fi
```

Paramètres

- *value* peut être de plusieurs types :
  - un entier;
  - un réel;
  - un caractère;
  - une chaîne de caractères;
  - un code d'erreur: SUCCESS / FAILURE.

Résultat

Retourne la valeur affectée.

Exemples

- Affecte la valeur SUCCESS :

```
psetstatus SUCCESS
pstatus {-> retourne la valeur SUCCESS }
```

- Affecte la valeur -12e20 :

```
psetstatus -12e20
pstatus {-> retourne la valeur -12e20 }
```

Voir aussi

Information, pstatus

Prototype C++

Aucun.

Auteur: Régis Clouard

psetsubband

Insertion d'une sous-bande dans une image de DWT.

Synopsis

psetsubband *scale subband* [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **psetsubband** insère une sous-bande dans une image DWT à l'échelle spécifiée *scale*. La sous-bande *im_in2* est une image et elle est ajouté à l'image d'entrée *im_in1*. A chaque échelle, les images sont numérotées ainsi:

```
[1][2]
[3][4]
```

- 1: sous-bande LL des coefficients d'approximation.
- 2: sous-bande LH des coefficients de détail.
- 3: sous-bande HL des coefficients de détail.
- 4: sous-bande HH des coefficients de détail.

Paramètres

- *scale* spécifie l'échelle d'analyse de l'image DWT.
- *subband* spécifie le numéro de la sous-bande à insérer à l'échelle donnée.

Entrées

- *im_in1*: une image 2D de niveaux de gris de DWT.
- *im_in2*: une image 2D image avec les bonnes dimensions.

Entrées

- *im_out*: une image de même type que l'image d'entrée *im_in1*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Seuille l'image LL issue de l'analyse par ondelette d'un carré.

```
pshapedesign 256 256 0 2 150 150 a.pan
pqmf daubechies 4 b.pan
pdwt 1 a.pan b.pan c.pan
pgetsubband 1 1 c.pan d.pan
pthresholding 20 400 d.pan e.pan
psetsubband 1 1 c.pan e.pan f.pan
pidwt 1 f.pan b.pan out.pan
```

Voir aussi

Domaine Fréquentiel

Prototype C++

```
Errc PSetSubband( const Img2dsf &im_in1, const Img2dsf &im_in2,
Img2dsf &im_out, int scale, int subband);
```

Auteur: Ludovic Soltys

pshapedesign

Création d'une image vierge ou avec une forme synthétique prédéfinie.

Synopsis

pshapedesign *width height depth type radius length* [*im_out*|-]

Description

L'opérateur **pshapedesign** crée une image de taille (*width, height, depth*) vide ou contenant une forme prédéfinie. C'est la valeur de *type* qui détermine le type de l'image et la forme synthétique à l'intérieur. Le fond est mis à 0, et la forme est mise à 255. Les formes sont orientées verticalement.

- Le centre de la forme est le centre de l'image.
- La hauteur, la largeur et la profondeur sont calculées à partir des valeurs des paramètres de rayon (*radius*) et la longueur (*length*).

Paramètres

- *width, height, depth* spécifient la taille de l'image de sortie, respectivement la longueur, la hauteur et la profondeur. Si la profondeur d=0 alors l'image de sortie est une image 2D. Si la hauteur h=0 alors l'image de sortie est une image 1D.
- *type* spécifie la forme:

  2D
  - 0- aucune forme
  - 1- disque
  - 2- carré
  - 3- rectangle

  3D
  - 10- aucune forme
  - 11- sphère
  - 12- cube
  - 13- parallélépipède
  - 14- cylindre
  - 15- bipyramide
- *radius* définit le diamètre d'une forme symétrique ou la hauteur et la profondeur d'une forme asymétrique.
- *length* définit la longueur d'une forme asymétrique. Pour les formes symétriques la valeur de *length* est ignorée mais doit être donnée.

Sorties

- *im_out*: une image de Uchar.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit une image synthétique (un carré) pour illustrer le phénomène de Gibbs en analyse par ondelettes.

```
pshapedesign 256 256 0 2 150 150 a.pan
pqmf daubechies 4 b.pan
pdwt 1 a.pan b.pan c.pan
psplitimage c.pan d1.pan d2.pan d3.pan d4.pan
pthresholding 20 400 d2.pan e2.pan
pthresholding 20 400 d3.pan e3.pan
pthresholding 20 400 d4.pan e4.pan
pmergeimages d1.pan e2.pan e3.pan e4.pan f.pan
pidwt 1 f.pan b.pan out.pan
```

Voir aussi

Utilitaire

Prototype C++

```
Errc PShapeDesign( const Img2duc &im_out, long type, int radius, int length );
```

Auteur: Jean-Marie Janik

psharp

Rehaussement du contraste par convolution.

Synopsis

psharp *connexity degree* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **psharp** effectue un rehaussement de contraste de l'image d'entrée *im_in* en utilisant le Laplacien. L'objectif du rehauseement de contraste est de rendre plus clairs les détails fins et de rendre plus nets les parties flous. Réhausser le contraste consiste à réduire la largeur de la transition de l'intensité dans affecté l'intensité moyenne de chaque coté de la variation.

Le principe du rehaussement de contraste utilisant le Laplacien est basé sur la soustraction de *degree* fois l'image d'entrée avec le Laplacien de l'image d'entrée. Il s'implante par un filtrage spatial avec le filtre suivant qui dépend de la valeur de connexité *connexity*).
Par exemple, les filtres 2D sont :

```
    4-connexité              8-connexité
 0      -1      0          -1     -1      -1
-1   4*degré -1    or     -1   8*degré -1
 0      -1      0          -1     -1      -1
```

En 3D, le centre du filtre est 6*degrée. 6-connexité ou 26*degrée. 26-connexité.

Paramètres

- *connexity* spécifie le type de connexité entre pixels voisins : 4 ou 8 pour les images 2D image - 6 ou 26 pour les images 3D.
- *degree* est une valeur réelle qui spécifie le degré de réhaussement. Plus le degré est élevé, moins l'effet du réhaussement est fort. Des valeurs typiques sont : 0.7, 1.0, 1.7, 2, 7.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image avec les mêmes propriétés que l'image d'entrée *im_in*.

Résultat

Retourne SUCCESS ou FAILURE en cas de mauvais paramétrage.

Exemples

Réhausse le contraste avec le laplacien (qui correspond à un degré 1).

```
psharp 8 1 tangram.pan a.pan
```

Voir aussi

Filtrage

Prototype C++

```
Errc PSharp( const Img2duc &im_in, Img2duc &im_out, int connexity,
float degree );
```

Auteur: Régis Clouard

pshen

Détection et localisation des contours de Shen-Castan.

Synopsis

pshen *strenght* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pshen** permet de localiser les contours de l'image *im_in*. L'image de sortie *im_out* est construite avec les valeurs d'amplitude maximales dans la direction du gradient. Le reste est mis à 0.

La valeur d'amplitude du gradient en un point reflète la variation de niveau de gris observé dans l'image *im_in* en ce point. Plus cette valeur est élévée plus cette variation est forte.

L'extraction se fait en trois étapes:

1. lissage ;
2. calcul du gradient en chaque point de l'image ;
3. extraction des maxima locaux avec ajustement des niveaux.

Paramètres

- *strength* donne l'intensité du lissage associé à la détection du gradient. Les valeurs sont typiquement dans l'intervalle [0..10]. Plus la valeur est faible, plus le lissage est fort et donc moins il y aura de contours. Une valeur de 0 correspond à un lissage total de l'image (donc il ne reste plus de gradient). Une valeur typique est 1.

Entrées

- *im_in*: une image.

Sorties

- *im_out*: une image du même type que l'image im_in.

Résultat

Retourne SUCCESS ou FAILURE.

Voir aussi

Détection de contours

Exemples

Détection de contours dans l'image tangram.pan:

```
pshen 1 tangram.pan a.pan
pbinarization 10 1e30 a.pan b.pan
```

Prototype C++

```
Errc PShen( const Img2duc &im_in, Img2duc &im_out, float strenght );
```

Auteur: Carlotti & Joguet

pshensmoothing

Lissage de Shen-Castan.

Synopsis

pshensmoothing *alpha* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pshensmoothing** permet de lisser une image par la méthode de Shen-Castan.

Paramètres

- *alpha* donne l'intensité du lissage associé à la détection du gradient. Les valeurs sont typiquement dans l'intervalle [0..10]. Plus la valeur est faible, plus le lissage est fort. Une valeur de 0 correspond à un lissage total de l'image (donc il ne reste plus de gradient). Une valeur typique est 1.

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Performs Shen smoothing on tangram.pan:

```
pshensmoothing 1 tangram.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PShenSmoothing( const Img2duc &im_in, Img2duc &im_out, float
alpha );
```

Auteur: Carlotti & Joguet

psig

Construction de la sphère d'influence d'un graphe.

Synopsis

psig [*-m mask*] [*gr_in*|-] [*gr_out*|-]

Description

L'opérateur **psig** consiste à modifier le *gr_in* en coupant les arcs dont les sommets ne sont pas dans la même sphère d'influence.

Les sphères d'influence sont des cercles centrés sur les sommets et dont les rayons sont les distances entre les deux sommets. Si les cercles de deux sommets voisins s'intersectent, l'arc est conservé, sinon l'arc est supprimé.

En sortie, les champs *value* sont remis à 1.

La distance entre deux sommets est calculée par la distance euclidienne des coordonnées de leur germe.

Entrées

- *gr_in*: un graphe.

Sorties

- *gr_out*: un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
psig g1.pan g2.pan
```

Voir aussi

Graphe

Prototype C++

```
Errc PSig( const Graph &gr_in, Graph &gr_out );
```

Auteur: François Angot

psigmafiltering

Lissage par filtre adaptatif basé sur le choix des voisins.

Synopsis

psigmafiltering *demitaille eps nbmin* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **psigmafiltering** effectue un lissage de l'image d'entrée. Le principe est de remplacer chaque point par la valeur moyenne de ses (demitaille*2+1) voisins.

On ne prend en compte dans la moyenne que les points, dont la différence avec le point central est inférieure à un intervalle donné *eps*. Si le nombre de point de la somme n'est pas suffisant (< *nbmin*), on remplace le point central par la somme de ses 4 voisins, sinon on prend la moyenne.

Paramètres

- *demitaille* permet de définir la fenêtre des voisins à prendre en compte dans la recherche de la valeur sigma.
- *eps* spécifie l'écart de la moyenne au point central au-dessous duquel la valeur peut être prise pour le calcul de la moyenne.
- *nbmin* permet de définir le nombre minimum de points utilisés pour le calcul de la moyenne.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applique un filtre sigma à l'image tangram.pan:

```
psigmafiltering 1 20 2 tangram.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PSigmaFiltering( const Img2duc &im_in, Img2duc &im_out, Short
demitaille, Ushort eps, Ushort nbmin );
```

Auteur: Régis Clouard

psimplelineariterativeclustering

Segmentation d'une image couleur en superpixels.

Synopsis

psimplelineariterativeclustering *sigma k minimum-region-area* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **psimplelineariterativeclustering** segmente l'image d'entrée par catégorisation des pixels dans l'espace des couleurs pour former des superpixels.

Paramètres

- *k* est le nombre de superpixels désiré. La valeur par défaut est 200.
- *m* est le facteur de compacité. C'est une valeur entre 10 et 40. La valeur par défaut est 10.

Entrées

- *im_out*: une image 2D.

Sorties

- *im_out*: une carte de régions.

Résultat

Retourne le nombre de regions ou FAILURE.

Exemples

Segmente les pièces de tangram:

```
psimplelineariterativeclustering 500 20 examples/tangram.pan a.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PSimpleLinearIterativeClustering( const Imc2duc &AMP;ims,
Imc2duc &AMP;imd, int k, float m );
```

Référence

"SLIC Superpixels", Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. EPFL Technical Report no. 149300, June 2010.

psizeselection

Sélection de régions sur leur taille (profondeur, hauteur et largeur).

Synopsis

psizeselection *relation width height depth* [*-m mask*] [*rg_in*|*-*] [*rg_out*|*-*]

Description

psizeselection sélectionne les régions par leur valeur de taille (depth, height and width). Le paramètre *relation* spécifie la relation d'ordre à la valeur de taille pour sélectionner ou pas une région.

Si une taille =-1 alors elle n'est pas prise en compte dans la sélection.

Paramètres

- *relation* est une valeur entière de l'intervalle [-2,2], précisant la relation à la valeur de taille:
  - *relation* = 2 : toutes les régions >= taille.
  - *relation* = 1 : toutes les régions > taille.
  - *relation* = 0 : toutes les régions = taille.
  - *relation* = -1 : toutes les régions < taille.
  - *relation* = -2 : toutes les régions <= taille.
- *depth* est un entier défini en pixels. Si la valeur de paramètre =-1 alors la taille n'est pas prise en compte.
- *height* est un entier défini en pixels. Si la valeur de paramètre =-1 alors la taille n'est pas prise en compte.
- *width* est un entier défini en pixels. Si la valeur de paramètre =-1 alors la taille n'est pas prise en compte.

Entrées

- *rg_in*: une carte de régions.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Exemples

Sélectionne les régions de largeur = 50 pixels dans la carte de régions rin.pan :

```
psizeselection 0 50 -1 0 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PSizeSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, long width, long height, long depth );
```

Auteur: Régis Clouard

pskeletonization

Squelettisation d'objets binaires.

Synopsis

pskeletonization *connexity* [*-m mask*] [*im_in*|-][*im_out*|-]

Description

L'opérateur **skeletonization** permet d'obtenir le squelette des objets binaires présents dans l'image *im_in*.

L'algorithme repose sur une succession d'amincissements jusqu'à obtention d'une structure stable ne pouvant être amincie, c'est à dire dont les éléments sont des lignes d'épaisseur 1 pixel.
L'amincissement est obtenu par 8 masques d'érosion dans les 8 directions possibles N, NO, O, SO, S, SE, E, NE. Un masque indique une forme possible d'une ligne selon la direction choisie. Par exemple le masque EST s'écrit:

```
s'il existe une telle configuration autour du pixel central
( x est 0 ou 1)
   0  x  1
   0  1  1
   0  x  1
alors le pixel central reste à 1, sinon il passe à 0.
```

Paramètre

- *connexity*: degré de connexité pour les objets {4 ou 8}.

Entrées

- *im_in*: une image binaire 2D.

Sorties

- *im_out*: une image binaire 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Création du squelette des pièces de tangram :

```
pbinarization 95 -1 examples/tangram.pan i1.pan
pskeletonization 4 i1.pan out.pan
```

Voir aussi

Morphologie

Prototype C++

```
Errc PSkeletonization( const Img2duc &im_in, Img2duc &im_out, int connexity );
```

Auteur: Régis Clouard

pskewanglecorrection

Correction de la déviation.

Synopsis

pskewanglecorrection *width height max_angle* [*-m mask*] [*rg1_in*|-] [*rg2_in*|-]

Description

L'opérateur **pskewanglecorrection** détecte and corrige la déviation dans un document en focntion de l'orientation de l'écriture.

Le masque de taille x est utilisé pour estimer l'angle de déviation. Le masque est extrait de l'image initiale et représente le meilleur compromis entre la densité et le nombre de transitions noires et blanches en colonne. Puis, l'angle de déviation est déterminé par l'angle qui donne le plus fort maximum dans la distribution horizontale de Wigner-Ville.

Paramètres

- *width, height* donne la taille du masque qui est utilisé pour estimer l'angle de déviation.
- *max_angle* est l'angle maxiumum accepté.

Entrées

- *im1_in*: une image.
- *im2_in*: une image.

Résultat

Retourne l'angle de déviation estimé.

Exemples

Voir aussi

Reconstruction

Prototype C++

```
Errc PSkewAngleCorrection( const Img2duc &rg1_in, const Img2duc
&rg2_in, int width, int height, int max_angle );
```

Auteur: Régis Clouard

psnnfiltering

Lissage par filtre adaptatif : Symetric Nearest Neighbourghood.

Synopsis

psnnfiltering [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **psnnfiltering** permet de lisser une image selon le principe du filtre adaptatif.

Le principe consiste à remplacer chaque point par la moyenne des points de valeur de moindre écart avec celle du centre par rapport à celui de son voisin opposé.

Le bord de l'image d'une taille de 1 pixel n'est pas affecté par le traitement.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applique un filtrage de type Symmetric Nearest Neighborhood filter à l'image tangram.pan :

```
psnnfiltering tangram.pan out.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PSnnFiltering( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Abderrahim Elmoataz

psnr

Calcul du rapport signal sur bruit.

Synopsis

psnr [*im_in1*|-] [*im_in2*|-]

Description

L'opérateur **psnr** mesure le rapport entre l'information et le bruit dans une image. Il est calculé à partir d'une iamge initiale *im_in1* qui contient le signal et le bruit et d'un eimage *im_in2* qui est la version restaurée ou améliorée de l'image initiale *im1_in*.
En conséquence, plus le SNR est élevé, meilleur est le signal et donc meilleur est le traitement de restauration ou d'amélioration.

Parce qu'il peut prendre une très grande plage de valeur, le SNR s'exprime avec une échelle logarithmique en décibel (dB).

Le SNR est défini comme suit :

$$S = 10 * \log_{10}(R_{12})$$
$$R_{12} = \text{sum}\{(ims1)^2\} / \text{sum}(ims2-ims1)^2\}$$

Les images d'entrée *im_in1* et *im_in2* doivent avoir la même dimension et le même type.

Pour les images couleur ou multispectrales, la définition du SNR est la même sauf que chaque somme utilise toutes les bandes.

Note: R_{12} est dépendant non seulement de la différence *ims1-ims2*, mais aussi de *ims1*. Ainsi, le rapport signal/bruit est dépendent des entrées et il est alors utilisable pour comparer des résultats à partir d'une même image initiale.

Entrées

- *im_in1*: une image.
- *im_in2*: une image (une version restaurée ou ameliorée version de im_in1).

Exemple

Calcule le SNR pour un filtre moyenneur:

```
pmeanfilter 2 tangram.pan i1.pan
psnr tangram.pan i1.pan
pstatus
```

Résultat

Retourne une valeur réelle positive exprimée en décibel dB.
(Utiliser `pstatus` pour récupérer cette valeur).

Voir aussi

Evaluation, pmse, ppsnr

Prototype C++

```
Errc PSNR( const Img2duc &im_in1, const Img2duc &im_in2 );
```

Auteur: Régis Clouard

psobel

Module du gradient de Sobel.

Synopsis

psobel [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **psobel** permet d'approximer le calcul de l'amplitude du gradient de l'image *im_out*.

L'algorithme consiste à convoluer l'image par le masque de Sobel:

```
|+1 +2 +1|
|+0 +0 +0|
|-1 -2 -1|
```

En fait ici, le masque est orienté successivement dans les quatre directions: 0, 45, 90, 135 degrés et c'est la valeur maximale qui est choisie comme amplitude.

L'image de sortie *im_out* est de même type que l'image d'entrée *im_in*.

Entrées

- *im_in*: une image.

Sorties

- *im_in*: une image du même type que l'image im_in.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Détection de contours dans l'image tangram.pan :

```
psobel tangram.pan b.pan
pbinarization 45 1e30 b.pan out.pan
```

Voir aussi

Détection de contours

Prototype C++

```
Errc PSobel( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

psphericityselection

Sélection de régions sur leur valeur de sphéricité.

Synopsis

psphericityselection *relation seuil* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **psphericityselection** permet de sélectionner les régions sur leur degré de sphéricité. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

La sphéricité est le rapport entre le rayon du cercle inscrit et le rayon du cercle circonscrit:

```
sphericite = rayon inscrit/rayon circonscrit.
```

Ce rapport vaut 1 pour un cercle.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur réelle [0..1] qui correspond au degré de sphéricité.

Entrées

- *rg_in*: une carte de régions 2D ou 3D.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Exemples

Sélectionne les régions avec le plus fort degré de sphéricité :

```
psphericityselection 3 0 a.pan b.pan
```

Voir aussi

Région

Prototype C++

```
Errc PSphericitySelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, float seuil );
```

Auteur: Régis Clouard

psplitimage

Eclatement d'une image en 4 sous-images.

Synopsis

psplitimage [*im_in*|-] [*im_out1*|-] [*im_out2*|-] [*im_out3*|-] [*im_out4*|-]

Description

L'opérateur **psplitimage** permet de générer 4 sous-images à partir d'une image origine. Si les images de sorties sont nommées 1, 2, 3,4, l'image d'entrée aura été éclatée comme suit:

```
[1][2]
[3][4]
```

La coupure est faite au milieu, donc si l'image n'est pas de taille paire, l'image du premier rectangle est plus petite que l'image du second rectangle.

Entrées

- *im_in*: une image 2D.

Sorties

- *im_out1*, *im_out2*, *im_out3*, *im_out4*: des images 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit une image synthétique pour illustrer le phénmène de Gibbs en analyse par ondelettes.

```
pshapedesign 256 256 0 2 150 150 a.pan
pqmf daubechies 4 b.pan
pdwt 1 a.pan b.pan c.pan
psplitimage c.pan d1.pan d2.pan d3.pan d4.pan
pthresholding 20 400 d2.pan e2.pan
pthresholding 20 400 d3.pan e3.pan
pthresholding 20 400 d4.pan e4.pan
pmergeimages d1.pan e2.pan e3.pan e4.pan f.pan
pidwt 1 f.pan b.pan out.pan
```

Voir aussi

Utilitaire, pmergeimages

Prototype C++

```
Errc PSplitImage( const Img2dsf &im_in, Img2dsf &im_out1, Img2dsf
&im_out2, Img2dsf &im_out3, Img2dsf &im_out4 );
```

Auteur: Ludovic Soltys

psqrt

Racine carrée d'une image ou d'un graphe.

Synopsis

psqrt [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **psqrt** construit la racine carrée d'une image. Chaque pixel de l'image de sortie *im_out* est construit avec la racine carrée du pixel correspondant dans l'image d'entrée *im_in*.

La formule de calcul est tout simplement:

```
pixel(im_out)=sqrt(pixel(im_in))
```

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

L'image de sortie est de type Float.

Pour les graphes, le graphe de sortie est construit avec la racine carrée des valeurs de noeuds.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *im_out*: une image de Floats ou un graphe.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
psqrt tangram.pan a.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PSqrt( const Img2duc &im_in, Img2duc &im_out );
```

Auteur: Régis Clouard

psquareintegralimage

Calcul de l’image intégrale carrée d’une image.

Synopsis

psquareintegralimage [*-m mask*] [*im_in*|-] [*col_out*|-]

Description

L’opérateur **psquareintegralimage** calcule l’image intégrale carrée de l’image d’entrée *im_in*. Le résultat est stocké dans un vecteur de taille nombre de lignes * nombre de colonnes de l’image d’entrée. L’image intégrale carrée permet de facilement calculer la somme carrée à l’intérieur d’une fenêtre de l’image.

L’image intégrale carrée est définie comme suit:

```
output(x,y) = SUM(input(i,j)*input(i,j)) où i dans [0..x] et j dans [0..y].
```

Entrées

- *im_in*: une image.

Sorties

- *col_out*: une collection avec un tableau de valeur nommé "internal_array".

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule l’image intégrale careée de tangram.pan.

```
psquareintegralimage tangram.pan d.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PsquareIntegralImage( const Img2duc &im_in, Collection
&col_out);
```

Auteur: Pierre Buyssens

pstatus

Affichage de la valeur retournée par la dernière exécution d'un opérateur Pandore.

Synopsis

`pstatus`

Description

L'opérateur **pstatus** affiche la valeur retournée par la dernière exécution d'un opérateur Pandore, à la manière de la commande echo $? du Shell.

Chaque opérateur Pandore retourne une valeur qui lui est propre. La documentation associée à l'opérateur décrit le type de la valeur retournée parmi:

- un entier;
- un réel;
- un caractère;
- une chaîne de caractères;
- un code d'erreur: SUCCESS / FAILURE.

Cette valeur peut ensuite être utilisée comme valeur de paramètre pour un autre opérateur :

- Sous UNIX, l'exécution de cette commande peut être utilisée pour affecter une variable :

```
plabeling 8 a.pan b.pan
a=‘pstatus‘
echo "Nombre de régions=%a%"
```

- Sous MSDOS, l'exécution de cette commande construit un fichier "pset.bat" qui peut ensuite être utilisé pour affecter la valeur de retour à une variable.
  Par exemple :

```
plabeling 8 a.pan b.pan
call pstatus
call pset a
echo "Nombre de régions=%a%"
```

Résultat

Ne retourne aucune valeur (Ce qui a pour conséquence que deux exécutions successives de **pstatus** affichent le même résultat!).

Exemples

- Unix/Linux/MACOS: affiche le nombre de régions qui résultent d'un processus de segmentation basée sur le seuillage par maximisation de la variance inter-classe :

```
pvariancebinarization examples/tangram.pan a.pan
plabeling 8 a.pan b.pan
a=‘pstatus‘
echo "Number of regions=%a%"
```

- MsDos: affiche le nombre de régions qui résultent d'un processus de segmentation basée sur le seuillage par maximisation de la variance inter-classe :

```
pvariancebinarization examples/tangram.pan a.pan
plabeling 8 a.pan b.pan
call pstatus
call pset a
echo "Number of regions=%a%"
```

Voir aussi

Information, psetstatus

Prototype C++

La valeur retournée est celle de la fonction appelée.

Auteur: Régis Clouard

pstereogram

Construction d'une image stéréogramme couleur.

Synopsis

pstereogram [*im_in1*|-] [*im_im2*|-] [*im_out*|-]

Description

L'opérateur **pstereogram** construit une image de stérégramme (faux 3D) à partir d'une image profondeur *im_in1* et d'un motif *im_in2*.

Un stéréogramme est une image dans laquelle des informations stéréoscopiques sont codées. Pour regarder un stéréogramme, il faut considérer que le plan focal se trouve derrière l'image.

Le motif est répéré avec déformation dans l'image destination de telle manière à épouser les formes de l'image de profondeur.

L'image de profondeur est une image de niveaux de gris, où la valeur de niveau de gris d'un pixel indique la profondeur du point correspondant.

L'image motif est une image couleur de petite taille. Cette image sera utilisée comme motif de base de l'image à construire.

Entrées

- *im_in1*: a 2D gray level image (Img2duc).
- *im_in2*: a 2D color image (Imc2duc).

Sorties

- *im_out*: a 2D color image (Imc2duc).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit l'image c.pan à partir de l'image de profondeur a.pan et l'image de motif b.pan. a.pan est construit à partir de l'image 3D a3d.pan:

```
pgraylevel2depth 50 a3d.pan a.pan
pstereogram a.pan b.pan c.pan
```

Voir aussi

Exotique

Prototype C++

```
Errc PStereogram( const Img2duc &imp, const Imc2duc &imt, Imc2duc
&im_out );
```

Auteur: Régis Clouard

psub

Soustraction d'images ou de graphes et différence non symétrique entre cartes de régions.

Synopsis

psub [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **psub** calcule la soustraction des valeurs de niveaux de gris entre chaque pixel de l'image *im_in1* et de l'image *im_in2* correspondant.

Il n'y a pas de gestion du débordement de valeurs. La formule reprend exactement l'opérateur du C :

```
pixel(im_out) = (pixel(im_in1) - pixel(im_in2).
```

Les deux images d'entrés *im_in1* ou *im_in2* doivent être de même type. Au besoin, il est nécessaire d'utiliser les opérateurs de coercition. Par contre, l'image de sortie est du type le plus grand possible par rapport au type des images d'entrée:

- Long entre images d'octets.
- Long entre images d'entiers.
- Float entre image de floats.

Pour les images couleur et multispectrale, l'opérateur est appliqué séparément sur chacune des bandes.

Pour les cartes de régions, la soustraction retourne une carte de régions construite avec les régions de *im_in1* privés des régions de *im_in2*. Elle correspond à l'opération ensembliste de différence non symétrique. La carte de région de sortie contient une nouvelle numérotation des labels.

Entrées

- *im_in1*: une image, un graphe ou une carte de régions.
- *im_in2*: une image, un graphe ou une carte de régions.

Sorties

- *im_out*: une image, un graphe ou une carte de régions

Résultat

Retourne SUCCESS ou FAILURE.

Pour les cartes de régions, retourne la valeur de label maximale.

Exemples

```
psub a.pan b.pan c.pan
```

Voir aussi

Arithmetique

Prototype C++

```
Errc PSub( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

psubsampling

Sous-échantillonnage d'une image

Synopsis

psubsampling *factor* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **psubsampling** permet de sous-échantilloner l'image d'entrée d'un facteur spécifié. Le sous-échantillonnage consiste à découper l'image en pavés de taille *factor* x *factor* et à mettre tous les pixels avec la couleur moyenne du pavé.

Paramètres

- *factor* est un entier positif qui indique la taille des pavés.

Entrées

- *im_in*: une image, une carte de régions ou un graphe.

Sorties

- *im_out*: une image de même type que l'objet d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Sous-échantillonnage d'un facteur 2 :

```
psubsampling 2 tangram.pan a.pan
```

Voir aussi

Miscellaneous.

Prototype C++

```
Errc PSubsampling( const Img2duc &im_in, Img2duc &im_out, int factor
);
```

Auteur: Régis Clouard

psubval

Soustraction d'une image avec des constantes stockées dans une collection.

Synopsis

psubval [*-m mask*] [*col_in*|-] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **psubval** calcule l'image *im_out* par soustraction des valeurs de pixels de l'image *im_in* avec les valeur stockes dans la collection *col_in*. La première valeur de la collection est soustraite à tous les pixels de la première bande, la seconde à tous les pixels de la seconde bande, etc.

Il y a écrétage du résultat si la valeur résultante est supérieure à la valeur maximale du type de l'image. La formule de calcul est la suivante :

```
val = pixel(im_in) - col_in;
if (val > MAX) pixel(im_out) = MAX;
else if (val < MIN) pixel(im_out) = MIN;
else pixel(im_out) = val;
```

Entrées

- *col_in*: une collection avec autant de valeurs réelles que de nombre de bandes pour l'image d'entrée (p. ex. 3 pour une image couleur).
- *im_in*: une image.

Sorties

- *im_out*: un objet du même type que *im_in*.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Retire a tangram.pan sa valeur moyenne:

```
pmeanvalue tangram.pan a.pan
psubval a.pan tangram.pan b.pan
```

Autres exemples

Voir aussi

Arithmetique

Prototype C++

```
Errc PSubVal( const Collection &col_in, const Img2duc &im_in,
Img2duc &im_out );
```

Auteur: Régis Clouard

psumvalue

Calcul de la somme des valeurs de pixels (ou de sommets).

Synopsis

psumvalue [*-m mask*] [*im_in*|-] [*col_out*|-]

Description

L'opérateur **psumvalue** calcule la somme des valeurs de pixels de l'image *im_in* ou des sommets s'il s'agit d'un graphe.

La valeur calculée est accessible par la commande **pstatus**.

Les valeurs de somme de chaque bande sont stockées dans la collection *col_out*.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *col_out*: une collection de valeurs réelles.

Résultat

Retourne la valeur réelle qui représente la somme totale de l'image *im_in* (pour la première bande uniquement).

La valeur peut être récupérée par l'opérateur **pstatus**.

Exemples

Mesure la somme globale de l'image tangram.pan (version Unix):

```
psumvalue tangram.pan col.pan
var=‘pstatus‘
echo "Somme = $val"
```

Mesure la somme globale de l'image tangram.pan (version MsDos):

```
psumvalue tangram.pan col.pan
call pstatus
call pset var
echo Somme = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Float PSumValue( const Img2duc &im_in, Collection & col_out );
```

Auteur: Régis Clouard

psuperimposition

Surimposition de contours sur une image.

Synopsis

psuperimposition *color_mask* [*im_in*|-] [*mk_in*|-] [*im_out*|-]

Description

La **psuperimposition** permet d'incruster des contours dans une image en choisissant la couleur d'affichage. Par exemple, dans le cas d'un image couleur si color_mask=0 alors les contours de l'image de *mk_in* sont ajoutés à la bande rouge de l'image initiale *im_in*.

Cet opérateur est généralement utilisé pour visualiser les contours extraits d'une image directement dans l'image elle-même.

L'image de sortie est du même type que l'image d'entrée.

Paramètres

- *color_mask* spécifie le masque de coloration. Il agit comme un masque de bits. Par exemple, si on considère une image initiale couleur:
  - mask=0: le masque est peint en noir.
  - mask=1: le masque est peint en rouge.
  - mask=2: le masque est peint en vert.
  - mask=3: le masque est peint en jaune : rouge+vert.
  - mask=4: le masque est peint en bleu.
  - mask=5: le masque est peint en violet : rouge+bleu.
  - mask=5: le masque est peint en magenta : vert+bleu.
  - mask=7: le masque est peint en blanc;

Entrées

- *im_in* : une image.
- *mk_in* : une image de niveaux de gris.

Sorties

- *im_out* : une image du même type que l'image initiale.

Exemples

- Visualise les contours extraits de l'image tangram dans l'image tangram :

```
pgradient 1 examples/tangram.pan i1.pan i2.pan
pbinarization 30 1e30 i1.pan i3.pan
pbinarization 60 1e30 i1.pan i4.pan
pgeodesicdilation 1 1 -1 i4.pan i3.pan i4.pan
psuperimposition 1 examples/tangram.pan i4.pan out.pan
```

Résultat

Retourne SUCCESS ou FAILURE.

Voir aussi

Visualization

Prototype C++

```
Errc PSuperimposition( const Imc2duc &im_in, const Img2duc &mk_in,
Imc2duc &im_out, int color_mask);
```

Auteur: Régis Clouard

psuperpixelsegmentation

Sur-segmentation d'une image en super-pixels.

Synopsis

psuperpixelsegmentation *compactness perturbation* [*-m mask*] [*reg_in*|-] [*im_in*|-] [*reg_out*|-][*im_out*|-]

Description

L'opérateur permet de construire une carte de régions, les superpixels, à partir d'une carte de germes *rg_in* et d'une image d'intensité *im_in*. Les germes sont des régions de taille 1. L'algorithme est basé sur une catégorisation des pixels à partir de l'équation Eikonale. L'image de sortie *im_out* contient les valeurs de distance géodésiques.

Le paramètre de compacité permet de régler la force de la compacité des superpixels.

Note : pour une image couleur, des résultats un peu meilleurs sont obtenus dans l'espace Lab.

Paramètres

- *compactness* est un paramètre de compacité qui influe sur la forme des régions obtenues. Les valeurs sont des réels entre 0 et 1. La valeur par défaut est 0.
- *perturbation* est un booléen qui indique s'il est possible de déplacer un peu les germes (*perturbation*=1) ou pas (*perturbation*=0).

Entrées

- *reg_in* : une carte de régions avec les germes initiaux.
- *im_in* : une image d'intensité.

Sorties

- *reg_out* : la carte des régions.
- *im_out* : une carte de distances geodesiques.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

On place des germes tous les 10 pixels en largeur et en hauteur, puis on fait une sursegmentation par superpixels :

```
./pseedplacement 10 10 0 examples/tangram.pan rgin.pan
./psuperpixelsegmentation 0 1 rgin.pan examples/tangram.pan rgout.pan geodesic-map.pan
pboundary 8 rgout.pan cont.pan
psuperimposition 3 cont.pan examples:tangram.pan result.pan
```

Voir aussi

Segmentation, pseedplacement

Prototype C++

```
Errc PSuperPixelSegmentation2D( const Img2duc &im_in, const Reg2d
&reg_in, Reg2d &reg_out, Img2duc &im_out, float compactness, int
perturb);
```

Auteur : Pierre Buyssens

psusan

Détection de points d'intérêt selon l'algorithme SUSAN.

Synopsis

psusan *threshold* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **psusan** permet de détecter les points d'intérêt dans l'image d'entrée *im_in*. Les points d'intérêt sont soit des coins en L, des jonctions en T, des jonctions en Y ou des points de forte variation de texture. Ils correspondent à des doubles discontinuités de la fonction d'intensité provoquées par des discontinuités de la fonction réflectance ou de profondeur.

Le principe de l'algorithme consiste en:

1. Placer un masque circulaire de rayon 3 autour de chaque pixel.
2. Calculer le nombre de pixels du masque ayant la même intensité que le centre.
3. Seuiller pour produire l'image de force de réponse du pixel.
4. Suppression des non-maxima pour ne garder que les coins.

L'image de sortie *im_out* est une image d'entiers Long qui code pour chaque pixel la force de la réponse.

Paramètres

- *threshold* détermine la différence maximale de niveaux de gris entre 2 pixels pour considérer qu'ils sont dans la même région. Plus *threshold* est grand, moins il y aura de coins. Une valeur typique est 20.

Entrées

- *im_in*: une image d'intensité 2D de Uchar.

Sorties

- *im_dest*: une image de Long.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Extrait les coins dans l'image tangram.pan et superimpose le résultats sur l'image initiale :

```
psusan 20 tangram.pan a.pan
pbinarization 1000 1e30 a.pan b.pan
padd b.pan tangram.pan out.pan
```

Voir aussi

Points d'intérêt

Prototype C++

```
Errc PSusan( const Img2duc &im_in, Img2dsl &im_out, int threshold );
```

Auteur: Régis Clouard

pthresholding

Seuillage d'une image selon la valeur de pixel.

Synopsis

pthresholding *seuilb seuilh* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pthresholding** consiste uniquement à mettre à 0, toutes les valeurs de pixel inférieures strictement au *seuilb* ou supérieures strictement au *seuilh* et à conserver les valeurs de pixel comprises entre ces deux seuils.

```
if im_in[p] ≥ low and im_in[p] ≤ high
then im_out[p]=im_in[p];
else im_out[p]=0;
```

Si *high* est inférieur à *low* alors **pthresholding** effectue le seuillage à l'envers:

```
if im_in[p] < high or im_in[p] > low
then im_out[p]=im_in[p];
else im_out[p]=0;
```

Pour les carte de régions, le seuillage consiste à sélectionner les régions de valeur de label supérieure ou égale au *seuilb* ou supérieure ou égale au *seuilh*. Il n'y a pas de réétiquetage des labels de la carte de sortie *im_out*.

L'image de sortie *im_out* est du même type que celle d'entrée *im_in*.

Paramètres

- Le *seuilb* et le *seuilb* sont des valeurs de niveaux de gris (ou couleur) appartenant aux valeurs supportées par le type de l'image (ex: Img2duc [0..255], Img2dsl [-2147483648..+2147483648]).

  Astuce: Si *seuilh* est supérieur à la valeur maximale du type des pixels alors c'est la valeur maximale qui est utilisée. (ex: 255 pour Img2duc, +2147483648 pour Img2dsl).

Entrées

- *im_in*: une image, une carte de régions ou un graphe.

Sorties

- *im_out*: un objet du même type que l'entrée.

Résultat

Retourne SUCCESS ou FAILURE en cas de mauvais paramétrage.

Exemples

Sélection des pixels des pièces de tangram:

```
pthresholding 100 1e30 tangram.pan out.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PThresholding( const Img2duc &im_in, Img2duc &im_out, Uchar
seuilb, Uchar seuilh );
```

Auteur: Régis Clouard

ptiff2pan

Conversion d'une image Tiff en une image Pandore.

Synopsis

ptiff2pan *im_in* [*im_out*|-]

Description

L'opérateur **ptiff2pan** transforme l'image *im_in* du format Tiff au format Pandore. L'image Pandore résultante *im_out* dépend bien évidemment du format de l'image tiff *im_in*. Le format du fichier Pandore résultant peut être obtenu par la commande **pfile**.

Attention: les images compressées ne sont pas prises en compte. Il est alors nécessaire d'utiliser un autre logiciel pour supprimer la compression.

Entrées

- *im_in*: un fichier TIFF.

Sorties

- *im_out*: une image Pandore 2D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
ptiff2pan image.tiff a.pan
```

Voir aussi

Conversion, ppan2tiff

Prototype C++

```
Img2duc* PTiff2Pan( const char *filename );
```

Auteur: Régis Clouard

ptranslation

Construction du translaté d'une image.

Synopsis

ptranslation *direction decalage* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ptranslation** permet de construire une nouvelle image *im_out* en translatant le contenu de l'image *im_in* de *decalage* pixels dans la *direction* indiquée.

Les nouvelles valeurs ajoutées sont égales à 0.

Paramètres

- *direction* donne la direction de translation:
  - 0 = translation en x.
  - 1 = translation en y.
  - 2 = translation en z.
- *decalage* est une valeur entière positive ou négative selon le sens du décalage souhaité.

Entrées

- *im_in*: une image ou une carte de régions.

Sorties

- *im_out*: une image de même type que l'image d'entrée ou une carte de régions.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Translate le contenu de l'image tangram.pan de 50 pixels à gauche:

```
ptranslation 0 50 tangram.pan a.pan
```

Voir aussi

pscrolling, Transformation

Prototype C++

```
Errc PTranslation( const Img3duc &im_in,Img3duc &im_out, Uchar
direction, Long decalage );
```

Auteur: Régis Clouard

ptransposition

Construit le transposé d'une image selon un axe.

Synopsis

ptransposition *direction* [*im_in*|-] [*im_out*|-]

Description

L'opérateur **ptransposition** construit le transposé de l'image *im_in* selon un axe. Le transposé correspond à un changement de repère d'une image, par exemple les lignes deviennent les colonnes et les colonnes des lignes.

Paramètres

- *direction* indique la direction de la transposition:
  - 0 = transposition x en y et y en x.
  - 1 = transposition x en z et z en x.
  - 2 = transposition y en z et z en y.

  Pour une image 2D, le paramètre est ignoré mais doit être donné. Il correspond à la direction 0.

Entrées

- *im_in*: une image.

Sorties

- *im_in*: une image de même type que l'image d'entrée.

Exemples

Transpose le contenu de l'image tangram.pan:

```
ptransposition 0 tangram.pan a.pan
```

Résultat

Retourne SUCCESS ou FAILURE.

Voir aussi

Transformation

Prototype C++

```
Errc PTransposition( const Img2duc &im_in, Img2duc &im_out, int
direction );
```

Auteur: Régis Clouard

ptxt2col

Conversion d’un fichier texte en une collection.

Synopsis

`ptxt2col` *`filename`* [*`col_out`*|`-`]

Description

L’opérateur **ptxt2col** permet de créer une collection contenant des valeurs et des tableaux.

Pour une valeur, le fichier doit avoir la forme:

```
Type nom valeur
```

Exemple:

```
Float pi 3.141592
```

Pour un tableau, le type est préfixé par `Array:`*`type`* et les valeurs sont multiples.

Exemple:

```
Array:Ushort iota 1 2 3 4 5 6 7
Array:Char hop 15 34 x 6 56 72 78
```

Cet exemple crée une collection contenant un tableau `iota` de 7 Ushort et une collection contenant un tableau `hop` de 10 Char. (`34 x 6` indique que 34 est répété 6 fois).

Cas particulier: les chaînes de caractères qui sont stockées dans un Array:Char et sont de la forme:

```
Array:Char nom valeur
```

Exemple:

```
Array:Char my_string hello_word
```

Les retours à la ligne jouent le même rôle que les espaces dans la séparation des mots.

Entrées

- *filename*: un fichier texte au bon format.

Sorties

- *col_out*: une collection.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Crée la collection col.pan à partir du fichier texte a.txt:

```
ptxt2col a.txt col.pan
```

Voir aussi

Collection

Prototype C++

```
Errc PTxt2Col( const std::string &filename, Collection &col_in_out
);
```

Auteur: Alexandre Duret-Lutz

ptxt2pan

Conversion d'une liste de points dans un fichier texte en une image.

Synopsis

ptxt2pan *type width height depth file_in* [*im_out*|-]

Description

L'opérateur **ptxt2pan** opérateur permet de fabriquer une image de taille (l colonnes, h lignes, p plans) et d'y placer les points contenus dans le fichier *file_in*. Le fichier de description *file_in*doit avoir la structure suivante:

```
valeur x y (z)
```

La ligne précédente place la valeur aux coordonnées (z,y,x). Cette disposition des coordonnées permet d'éliminer *z* (s'il existe) pour construire une image 2D, à partir d'un fichier 3D.

Paramètres

- *type* définit le type d'image à construire:
  - 0 : Img2duc
  - 1 : Img2dsl
  - 2 : Img2dsf
  - 3 : Img3duc
  - 4 : Img3dsl
  - 5 : Img3dsf
- *width*, *height* et *depth* spécifient respectivement le nombre de colonnes, de lignes et de plans de la future image.

Entrées

- *file_in*: un fichier texte au bon format.

Sorties

- *im_out*: une image.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Construit une image 2D d’octets de taille 256x256 avec les pixels donnés dans le fichier image.txt :

```
ptxt2pan 0 256 256 0 image.txt image.pan
```

Voir aussi

Conversion, ppan2txt

Prototype C++

```
Errc PTxt2pan( const Img3duc &im_out, char *nom, Long depth, Long height, Long width );
```

Auteur: François Angot

puniformitymerging

Fusion prioritaire de régions selon le critère d'uniformité.

Synopsis

puniformitymerging *nb_fusion seuil* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

L'opérateur **puniformitymerging** permet de fusionner les régions de la carte de régions *rg_in* selon le critère de l'uniformité.

La notion de voisinage entre les régions est détenue par le graphe *gr_in*.

Le principe de l'algorithme est le suivant:
Pour chaque région de la carte de régions *rg_in*, l'algorithme calcule la différence de variance avec la région voisine. Si la différence est inférieure au *seuil* donné en paramètre, alors les régions sont fusionnées.
On utilise ici l'algorithme de croissance prioritaire qui consiste à fusionner à chaque fois les 2 régions dont la différence est la plus faible.

L'uniformité est calculée par la formule:

```
uniformite(R)= (1-variance(R)/moyenne(R)^2) des deux régions).
```

Paramètres

- *nb_fusion* permet de spécifier le nombre de fusion à effectuer (la valeur -1 signifie d'ignorer ce paramètre et d'exécuter l'algorithme tant qu'il y a des fusions possibles).
- *seuil* permet de spécifier la tolérance par rapport au critère d'uniformité entre deux régions. Les valeurs appartiennent à l'intervalle [0..1], où 1 correspond à des régions totalement uniformes. On utilisera en général des valeurs proches de 1 (e.g., 0.95).

Entrées

- *rg_in*: les cartes de régions.
- *gr_in*: les graphes.

Sorties

- *rg_out*: une carte de régions.
- *gr_out*: un graphe.

Résultat

Retourne le nombre de fusions effectuées.

Exemples

Fusionne les régions issues d'une partition :

```
puniformityquadtree 0.9 tangram.pan a.pan
prg2gr a.pan b.pan
puniformitymerging -1 0.99 a.pan b.pan tangram.pan c.pan d.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PUniformityMerging( const Reg2d &rg_in, const Graph2d &gr_in,
Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, Uchar seuil );
```

Auteur: Laurent Quesnel

puniformityquadtree

Segmentation d'une image par quadtree (ou octree) selon l'uniformité.

Synopsis

puniformityquadtree *seuil* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **puniformityquadtree** permet de segmenter l'image en différentes régions selon le critère de l'uniformité. Les régions obtenues sont rectangulaires.

Le principe de l'algorithme est le suivant:

- Si un bloc n'est pas homogène (i.e. l'uniformité est inférieure au seuil) alors on le divise en 4 blocs égaux et on réapplique l'algorithme sur chacun des blocs.

On utilise ici la valeur de l'uniformitégalculée par:

```
uniformite(R)= 1 - ( variance(R) / (moyenne(R)^2) )
```

Paramètres

- *seuil* est la valeur d'uniformité minimum considérée pour qu'une région soit acceptée comme uniforme. Les valeurs appartiennent à l'intervalle [0..1]:
  - 1 correspond à des régions très uniformes.
  - 0 correspond à des régions peu uniformes.

  Généralement on utilise la valeur *seuil*=0.999 ce qui permet de jauger la précision.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *rg_in*: une carte de régions.

Résultat

Retourne le nombre de régions obtenues.

Exemples

Construit une partition de tangram.pan :

```
puniformityquadtree 0.9 tangram.pan a.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PUniformityQuadtree( const Img2duc &im_in, Reg2d &rg_out, float seuil );
```

Auteur: Laurent Quesnel

pvalueclassnumber

Comptage du nombre de valeurs différentes dans une image, une carte de régions ou un graphe.

Synopsis

pvalueclassnumber [*-m mask*] [*im_in*|-]

Description

L'opérateur **pvalueclassnumber** compte le nombre de valeurs différentes dans l'image d'entrée *im_in*. La valeur 0 est comptée comme une valeur normale.

Pour une carte de régions, l'opérateur compte le nombre de labels réellement prédents. *Attention*: le label 0 n'est pas comptée comme une valeur. Ce nombre peut différer du nombre de labels affiché par la carte de régions si tous les labels ne sont pas utilisés (i.e., des régions avec une surface nulle).

Pour un graphe, **pvalueclassnumber** compte le nombre de sommets de valeurs différentes (*champ value*). La valeur 0 est comptée comme une valeur normale.

Entrées

- *im_in*: une image, une carte de régions ou un graphe.

Résultat

Retourne la valeur entière qui représente le nombre de valeurs différentes dans l'image *im_in*. Cette valeur est accessible par la commande **pstatus**.

Exemples

Retourne le nombre de niveaux de gris utilisés dans tangram.pan:

```
pvalueclassnumber tangram.pan
pstatus
```

Voir aussi

Caractérisation image

Prototype C++

```
Double PValueClassNumber( const Img2duc &im_in );
```

Auteur: Régis Clouard

pvaluenumber

Comptage du nombre de pixels non nuls dans une image (un graphe ou une carte de régions).

Synopsis

pvaluenumber [*-m mask*] [*im_in*|-]

Description

L'opérateur **pvaluenumber** compte le nombre de pixels non nuls de l'image *im_in*, de sommets non nuls s'il s'agit d'un graphe ou de labels non nuls dans une carte de régions.

Entrées

- *im_in*: une image, une carte de région ou un graphe.

Résultat

Retourne la valeur entière qui représente le nombre total de pixels non nuls dans l'image *im_in*.
Cette valeur peut être récupérée par l'opérateur **pstatus**.

Exemples

Retourne le nombre de pixels non nuls dans l'image tangram.pan (version Unix):

```
pvaluenumber tangram.pan
val=‘pstatus‘
echo "Total = $val"
```

Retourne le nombre de pixels non nuls dans l'image tangram.pan (version MsDOS):

```
pvaluenumber tangram.pan
call pstatus
call pset val
echo Total = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Double PValueNumber( const Img2duc &im_in );
```

Auteur: Régis Clouard

pvaluerank

Détermination de la ieme valeur d'une image.

Synopsis

pvaluerank *index* [*-m mask*] [*im_in*|*-*]

Description

L'opérateur **pvaluerank** permet d'obtenir la *index* ième valeur (en commençant par 1) présente dans une image ou dans les sommets d'un graphe.

C'est un opérateur particulièrement utile pour connaître la valeur d'un seuil d'une image obtenue par classification. En effet, après la classification, l'image de sortie est construite avec comme label des régions la valeur du seuil utilisée pour classifier cette région (Voir les opérateurs de classification).

Paramètres

- *index* est un entier entre [1..MAX+1], où dépend du type de l'image d'entrée (e.g., 255 pour une image de char). Si sa valeur est supérieure à la dernière valeur, alors c'est cette dernière qui est retournée.

Entrées

- *im_in*: une image de niveaux de gris, une carte de régions ou un graphe.

Résultat

Retourne une valeur de pixel ou FAILURE si la *index*eme valeur n'existe pas.
Cette valeur peut être récupérée par l'opérateur **pstatus**.

Exemples

Retourne la valeur de seuil obtenue par un seuillage selon Chanda de l'image tangram.pan (version Unix)

```
pchanda 10 tangram.pan a.pan
pvaluerank 1 a.pan
seuil=‘pstatus‘
```

Retourne la valeur de seuil obtenue par un seuillage selon Chanda de l'image tangram.pan (version MsDos)

```
pchanda 10 tangram.pan a.pan
pvaluerank 1 a.pan
call pstatus
call pset seuil
```

Voir aussi

Extraction caractéristiques image

Prototype C++

```
Errc PvalueRank( const Img3duc &im_in, int index );
```

Auteur: Régis Clouard

pvarianceaggregation

Croissance des régions d'une carte selon la variance intérieure.

Synopsis

pvarianceaggregation *connexite seuil* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pvarianceaggregation** consiste à agglomérer des pixels à une région connexe lorsque sa valeur de pixel est proche de celle de la région (ie, la différence entre la variance de la région et la valeur de la variance de la région + ce pixel <= seuil).

Les pixels à agglomérer sont les pixels non encore étiquétés dans la carte de régions *rg_in* (ceux qui ont un label=0).

On agglomère un pixel à une région connexe si:

```
|variance(R)-variance(R+im_in[p])| <= seuil
```

La variance des régions de *rg_in* n'est pas recalculée pour éviter de trop s'éloigner de la situation initiale.

On préferera des exécutions itératives de cet opérateur. On pourra pour exemple itérer cet opérateur jusqu'à ce que le résultat de pstatus = 0. Ainsi, à chaque appel de l'opérateur la variance est recalculée avec les nouvelles régions.

La carte de sortie *rg_out* a le même nombre de labels que la carte d'entrée *rg_in*.

Paramètres

- La *connexite* est lié à la dimension l'image 4 ou 8 pour le 2D, et 6 ou 26 pour le 3D.
- Le *seuil* fixe l'écart toléré à la variance d'une région pour y agglomérer un pixel. C'est une valeur réelle de l'intervalle [0..niveau de gris^2] qui est égale au carré de l'écart-type.

Entrées

- *rg_in*: une carte de régions.
- *im_in*: une image de niveaux de gris.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre total de pixels qui ont été aggrégés à une région. Retourne FAILURE en cas de problème.

Exemples

Aggrège les pixels des pièces de tangram :

```
pbinarization 96 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pvarianceaggregation 8 1 b.pan tangram.pan out.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PVarianceAggregation( const Reg2d &rg_in, const Img2duc &im_in,
Reg2d &rg_out, int connexite, Uchar seuil );
```

Auteur: Régis Clouard

pvariancebinarization

Binarisation de l'image par analyse de la variance interclasse selon l'algorithme de Otsu.

Synopsis

pvariancebinarization [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pvariancebinarization** permet de binariser l'image initiale *im_in* selon une méthode basée sur l'analyse de l'histogramme. La valeur de seuil est prise comme celle qui maximise la variance interclasse.

Cet opérateur est basé sur l'algorithme de Otsu:
Principe: soit h l'histogramme des niveaux de gris, pi la probabilité pour qu'un pixel ait le niveau de gris i, et N le nombre de pixels total :

```
pi=h[i]/N
```

Pour diviser l'image en 2 classes C0 et C1 avec le seuil s:
On définit la probabilité pour un pixel x d'appartenir à la classe C0 puis à C1 par :

```
p(C0) = sum(i=0;s){ pi }
p(C1) = sum(i=s+1;N){ pi }
```

On définit la moyenne des classes C0 et C1 par :

```
M0 = sum(i=0;s){ pi/p(C0) }
M1 = sum(i=s+1;N){ pi/p(C1) }
```

On définit la moyenne totale de l'image d'entrée par :

```
Mt = sum(i=0;N){ i.pi }
```

Le seuillage automatique optimal consiste à trouver le seuil s qui maximise la variance interclasse V(s) telle que :

```
V(s) = sum(i1;2) { P(Ci).(Mi-Mt).(Mi-Mt) }
```

L'image de sortie *im_out* est contruite par binarisation.

Entrées

- *im_in*: une image de niveaux de gris 2D ou 3D.

Sorties

- *im_out*: une image d’octets.

Résultat

Retourne la valeur de seuil calculée.

Exemples

Segmente l’image tangram.pan:

```
pvariancebinarization tangram.pan a.pan
```

Voir aussi

Seuillage

Prototype C++

```
Errc PVarianceBinarization( const Img2duc &im_in, Img2duc &im_out );
```

Référence

N. Otsu, "A threshold selection method from grey scale histogram", *IEEE Trans. on Syst. Man and Cyber.*, vol 1, pp 62-66, 1979.

Auteur: Régis Clouard

pvariancefiltering

Filtrage d'une image par variance.

Synopsis

pvariancefiltering *seuil_bas seuil_haut* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pvariancefiltering** permet de filtrer l'image d'entrée *im_in*, par application d'un filtre de variance. Chaque pixel est remplacé par la variance de ses 8 voisins en 2D ou de ses 26 voisins en 3D si elle est comprise entre les deux seuils.

Paramètres

- *seuil_bas* et *seuil_haut* donnent les valeurs acceptables pour le remplacement et sont proportionnelles à la variance des valeurs de pixels.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *im_out*: une image de réels (Img2dsf ou Img3dsf).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Applique le filtre de variance à l'image tangram.pan:

```
pvariancefiltering 0 255 tangram.pan a.pan
```

Voir aussi

Filtrage spatial

Prototype C++

```
Errc PVarianceFiltering( const Img2duc &im_in, Img2duc &im_out,
Uchar seuil_bas, Uchar seuil_haut );
```

Auteur: François Angot

pvariancemerging

Fusion prioritaire de régions selon le critère de la variance.

Synopsis

pvariancemerging *nb_fusion seuil* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

L'opérateur **pvariancemerging** permet de fusionner les régions de la carte de régions *rg_in* selon le critère de la variance.

La notion de voisinage entre les régions est détenue par le graphe *gr_in*.

Le principe de l'algorithme est le suivant:
Pour chaque région de la carte de régions *rg_in*, l'algorithme calcule la différence de variance avec la région voisine. Si la différence est inférieure au *seuil* donné en paramètre, alors les 2 régions sont fusionnées.
On utilise ici l'algorithme de croissance prioritaire qui consiste à fusionner à chaque fois les 2 régions dont la différence est la plus faible.

La variance est calculée par:

```
variance(R)=moment2(R)-moyenne(R)^2
où moment2(R) = moment d'ordre 2 de la région R.
```

Paramètres

- *nb_fusion* spécifie le nombre de fusion à effectuer (la valeur -1 fait exécuter l'algorithme jusqu'à idempotence).
- *seuil* donne la valeur maximale acceptée comme différence entre les variances pour la fusion de 2 régions. Les valeurs appartiennent à l'intervalle [0..nombre de niveaux de gris^2] et correspondent au carré de l'écart-type.

Entrées

- *rg_in*: une carte de régions.
- *gr_in*: un graphe.
- *im_in*: une image.

Sorties

- *rg_out*: une carte de régions.
- *gr_out*: un graphe.

Résultat

Retourne le nombre de fusions effectuées.

Exemples

Fusionne les régions retournées par le processus de division de image tangram:

```
puniformityquadtree 0.9 examples/tangram.pan a.pan
prg2gr a.pan b.pan
pvariancemerging -1 45 a.pan b.pan examples/tangram.pan c.pan d.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PVarianceMerging( const Reg2d &rg_in, Graph2d &gr_in, Img2duc
&im_in, Reg2d &rg_out, Graph2d &gr_out, double nb_fusion, Uchar
seuil );
```

Auteur: Laurent Quesnel

pvariancequadtree

Segmentation d’une image par quadtree selon la variance.

Synopsis

pvariancequadtree *seuil* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

L’opérateur **pvariancequadtree** permet de segmenter l’image en différentes régions selon le critère de la variance.

Le principe de l’algorithme est le suivant:

- Si un bloc n’est pas homog`ne (i.e. la variance est supérieure au *seuil*) alors on le divise en 4 blocs égaux et on réapplique l’algorithme sur chacun des blocs.

Les régions obtenues dans la carte de régions de sortie *rg_out* seront donc rectangulaires.

On utilise ici la valeur de la variance calculée par:

```
variance(R)= SOMME((im_in[i] - moyenne(R))^2, i in R) / N
où im_in[i] sont tous les pixels de la région R
et N est le nombre de pixels de l’image
```

Paramètres

- *seuil* est la valeur de variance maximale pour qu’une région soit acceptée comme uniforme. Les valeurs appartiennent à l’intervalle des valeurs de niveau de gris possibles de l’image *im_in*.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne le nombre de régions obtenues.

Exemples

Construit une partition de l’image tangram.pan:

```
pvariancequadtree 10 tangram.pan a.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PVarianceQuadtree( const Img2duc &im_in, Reg2d &rg_out, float
seuil );
```

Auteur: Laurent Quesnel

pvarianceselection

Sélection de régions sur leur valeur de variance.

Synopsis

pvarianceselection *relation seuil* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

L'opérateur **pvarianceselection** permet de sélectionner les régions sur leur variance. Le paramètre *relation* spécifie la relation d'ordre par rapport à *seuil* pour sélectionner ou non une région.

La variance d'une région est calculée par la formule :

```
var = ((n*sigma2)-(sigma*sigma)) / (n*n)
```

où sigma est la somme des valeurs de niveaux de gris de la région,
où sigma2 est la somme des carrés des valeurs de niveaux de gris de la région et
où n est le nombre de pixels de la région.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur réelle appartenant à l'intervalle des valeurs de variance supportées par l'image d'entrée.

Entrées

- *rg_in*: une carte de régions
- *im_in*: une image de niveaux de gris 2D ou 3D.

Sorties

- *rg_out*: une carte de régions

Résultat

Retourne le nombre de régions ainsi sélectionnée.

Exemples

Sélectionne les régions avec une variance ≤ 20:

```
pvarianceselection -2 20 rin.pan rout.pan
```

Voir aussi

Région

Prototype C++

```
Errc PVarianceSelection( const Reg2d &rg_in, Img2duc &im_in, Reg2d &rg_out, int relation, float seuil );
```
hr>

Auteur: Régis Clouard

pvariancevalue

Calcul de la valeur variance des pixels d'une image (d'un graphe ou d'une carte de régions).

Synopsis

pvariancevalue [*im_in*|-] [*col_out*|-]

Description

L'opérateur **pvariancevalue** retourne la valeur de variance des valeurs des pixels de l'image *im_in*, ou des sommets s'il s'agit d'un graphe.

La mesure de variance est faite selon la formule:

```
variance = MOMENT_2 - (MOYENNE * MOYENNE);
```

Remarque : Cet opérateur n'est pas masquable.

Les valeurs de variane de chaque bande sont stockées dans la collection *col_out*.

Entrées

- *im_in*: une image ou un graphe.

Sorties

- *col_out*: une collection de valeurs réelles.

Résultat

Retourne la valeur réelle qui représente la variance totale de l'image *im_in* (pour la première bande uniquement). Cette valeur est accessible par la commande **pstatus**.

Exemples

Mesure la variance globale dans l'image tangram.pan (version Unix):

```
pvariancevalue tangram.pan col.pan
val=‘pstatus‘
echo "Variance = $val"
```

Mesure la variance globale dans l'image tangram.pan (version MsDos):

```
pvariancevalue tangram.pan col.pan
call pstatus
call pset val
echo Variance = %val%
```

Voir aussi

Caractérisation image

Prototype C++

```
Float PVarianceValue( const Img2duc &im_in, Collection & col_out );
```

Auteur: Régis Clouard

pversion

Affichage du numéro de version de la distribution Pandore.

Synopsis

`pversion`

Description

L'opérateur **pversion** permet d'afficher le numéro de version de la distribution Pandore. Le résultat est sous la forme:

```
PANDORE 6.0.0 (2006-04-13)
```

Exemples

- Affiche la version courante de Pandore :

  ```
  pversion
  ```

Résultat

Pas de valeur de retour.

Voir aussi

Information

Auteur: Régis Clouard

pvff2pan

Conversion d'une image VFF en format Pandore 3D.

Synopsis

pvff2pan *im_in* [*im_out*|-]

Description

L'opérateur **pvff2an** convertit une image au format VFF (SunVision) en une image au format Pandore 3D.

Le type de l'image de sortie *im_out* st toujours de type 3D Uchar (Img3duc).

Entrées

- *im_in*: un fichier VFF.

Sorties

- *im_out*: une image Pandore 3D.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l'image image.vff en image.pan.

```
pvff2pan image.vff image.pan
```

Voir aussi

Conversion

Prototype C++

```
Errc Vff2Pan( const FILE *fp, Img3duc &im_out );
```

Auteur: François Angot

pvinet

Calcul de la mesure de dissimilarité entre deux segmentations basée sur le nombre de pixels mal segmentés.

Synopsis

pvinet [*-m mask*] [*rg1_in*|-] [*rg2_in*|-]

Description

L'opérateur **pvinet** calcule une mesure de dissimilarité entre une segmentations donnée dans la carte de régions *rg1_in*> et une carte de region de référence rg2_in (vérité terrain), telle que définie par Laurent Vinet[*].

La méthode consiste à déterminer toutes les paires de régions qui ont un maximum de recouvrement et de définir la mesure comme le nombre de pixels qui ne participent pas au recouvrement.

Le résultat est une valeur entre [0..1]. Plus le critère est petit, meilleure est la ressemblance.

Attention: Les régions de label=0 ne sont pas prises en compte pour la mesure.

Entrées

- *rg1_in*: une carte de région (une image segmentée.).
- *rg2_in*: une carte de région (image de référence).

Résultat

Retourne une valeur réelle positive entre [0,1]. 0 signifie que les régions sont identiques et 1 qu'aucune région ne se recouvre. (Utiliser `pstatus` pour récupérer cette valeur).

Exemples

Calcule la mesure de dissimilarité entre les deux cartes de région dont l'une est le translaté de l'autre.

```
pbinarization 80 1e30 tangram.pan i1.pan
plabeling 8 i1.pan i2.pan
ptranslation 0 10 i2.pan i3.pan
pvinet i2.pan i3.pan
pstatus
```

Voir aussi

Evaluation

Prototype C++

```
Errc PVinet( const Reg2d &rg1_in, const Reg2d &rg2_in );
```

Reference

[*] JP. Cocquerez, S. Philipp, "*Analyse d'images: filtrage et segmentation*", Masson, 1995.

Auteur: Régis Clouard

pvisu

Affichage d'un fichier de type Pandore, png, jpeg ou bmp.

Synopsis

pvisu [*options*] [*-m mask*] [*im_in*|-]

Description

L'opérateur **pvisu** permet d'afficher le contenu d'un fichier au format Pandore, png, jpeg ou bmp.
Tous les types de fichier Pandore sont visualisables :

- les images,
- les cartes de régions,
- les graphes,
- et les collections.

Pour dessiner des lignes ou des points, il faut utiliser l'opérateur pdraw.

Entrées

- *im_in*: une image, une carte de régions, un graphe ou une collection.

Paramètres

- *-nofork* permet de rendre pvisu bloquant jusqu'à sa fin.

```
pvisu -nofork tangram.pan
```

- toutes les options QT (voir documentation Qt). Par exemple:

```
pvisu -style -nofork motif tangram.pan
```

Résultat

Retourne le numéro du processus (PID) ou FAILURE.

Exemples

- Visualisation de l'image tangram.pan :

```
pvisu tangram.pan
```

- Visualisation du contenu du centre de l'image tangram.pan (size 50x50) :

```
pshapedesign 256 256 0 1 50 50 a.pan
pim2rg a.pan m.pan
pmask examples/tangram.pan m.pan | pvisu
```

Voir aussi

Visualisation, pdraw, pcontentsdisplay

Auteur: Régis Clouard

pvolumeselection

Sélection de régions sur leur valeur de volume.

Synopsis

pvolumeselection *sens seuil* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

L'opérateur **pvolumeselection** permet de sélectionner les régions sur la valeur de volume occupé. Le paramètre *relation* spécifie la relation d'ordre à la valeur de *seuil* pour sélectionner ou non une région.

La valeur de volume est calculée en nombre de pixels inclus dans la région et sur la frontière.

Paramètres

- *relation* est une valeur entière de l'intervalle [-3,3], précisant la relation à la valeur de *seuil*:
  - *relation* = 3 : les régions > de valeur maximale.
  - *relation* = 2 : toutes les régions >= *seuil*.
  - *relation* = 1 : toutes les régions > *seuil*.
  - *relation* = 0 : toutes les régions = *seuil*.
  - *relation* = -1 : toutes les régions < *seuil*.
  - *relation* = -2 : toutes les régions <= *seuil*.
  - *relation* = -3 : les régions > de valeur minimale.
- Le *seuil* est une valeur entière en nombre de pixels.

Entrées

- *rg_in*: une carte de régions 3D.

Sorties

- *rg_out*: une carte de régions 3D.

Résultat

Retourne le nombre de régions ainsi sélectionnées.

Voir aussi

Région

Exemples

Sélectionne les régions avec un volume égal à 50 voxels.

```
pvolumeselection 0 50 reg1.pan reg2.pan
```

Prototype C++

```
Errc PVolumeSelection( const Reg3d &rg_in, Reg2d &rg_out, int
relation, Ulong seuil );
```

Auteur: Régis Clouard

pvoronoi

Calcul de la partition de Voronoï.

Synopsis

pvoronoi [*-m mask*] [*rg_in*|-] [*rg_out*|-][*im_out*|-]

Description

L'opérateur **pvoronoi** construit le diagramme de Voronoï à partir de la carte des germes *rg_in*. Un germe est un pixel ayant une valeur de label unique repéré dans une carte de régions.

Le diagramme est l'ensemble des polygones de Voronoï. On appelle polygone de Voronoï associé du germe Pi la région Vor(Pi) (chaque région étant l'ensemble de points (x,y) les plus proches à un point de P) telle que chaque point de P a pour plus proche germe Pi.

rg_out est la carte des régions de Voronoï résultantes, où les régions conservent la même numérotation que les germes correspondants.

im_out est l'image de distance associée au graphe de Voronoï. La distance utilisée est celle de chanfrein.

Entrées

- *rg_in*: une carte de région.

Sorties

- *rg_out*: une carte de régions.
- *im_out*: une image du même type que l'image d'entrée.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Calcule le diagramme de Voronoï èpartir des centres de gravités des pièces de tangram :

```
pbinarization 96 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pfillhole b.pan c.pan
pcenterofmass c.pan d.pan
pvoronoi d.pan out1.pan out2.pan
```

Voir aussi

Segmentation

Prototype C++

```
Errc PVoronoi( Reg &im_in, Reg2d &rg_out1, Img2dsl &im_out );
```

Auteur: Régis Clouard

pwatershed

Ligne de Partage des Eaux.

Synopsis

pwatershed [*-m mask*] [*rg_in*|-] [*im_pot*|-] [*rg_out*|-]

Description

L'opérateur **pwatershed** segmente les images en régions en utilisant une ligne de partage des eaux à partir de germes initiaux. La ligne de partage des eaux permet de faire croître les germes donnés dans la carte d'entrée *rg_in* en utilisant l'ordre de priorité donné par l'image de potentiels *im_pot*. La carte de sortie *rg_out* garde les mêmes valeurs de label que la carte d'entrée *rg_in*.

Si l'image de potentiels est une image de niveaux de gris, chaque valeur de pixel correspond à la valeur de potentiel du point. Par exemple, une image de distance aux frontières des objets ou directement l'image des niveaux de gris sont des images de potentiels acceptables.
Dans ce cas, le principe de l'algorithme est d'étiqueter tous les pixels qui touchent une région germe en commencant par ceux qui ont la valeur de potentiel la plus basse. Pour cela, on gère une file prioritaire.

Si l'image de potentiels est une image couleur, alors la valeur de potentielle est la distance euclidienne entre la couleur du point et la couleur moyenne de la région.

Remarque: Pour obtenir un résultat acceptable, il faut que les germes soient dans des puits de potentiels. Pour cela, il peut être nécessaire d'inverser l'image de potentiels (voir pinverse).

Entrées

- *rg_in*: une carte de régions.
- *im_pot*: une image de niveaux de gris ou de couleur.

Sorties

- *rg_out*: une carte de régions.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

- Squelette par zone d'influence (le skiz):

```
pbinarization 100 1e30 examples/tangram.pan i1.pan
pdistance i1.pan i2.pan
plabeling 8 i1.pan i3.pan
pwatershed i3.pan i2.pan i4.pan
pboundary 8 i4.pan out.pan
```

Voir aussi

Morphologie

Prototype C++

```
Errc PWatershed( const Reg2d &rg_in, const Img2duc &im_pot, Reg2d
&rg_out );
```

Auteur: Abderrahim Elmoataz, Olivier Lezoray

pweszka

Multiseuillage de l'image par analyse de la matrice de co-occurence selon Weszka.

Synopsis

pweszka *length* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pweszka** permet de multiseuiller l'image initiale *im_in* par classification des pixels selon l'algorithme de Weszka.

Cet algorithme est basé sur le calcul de la matrice de co-occurences Tkl symétrique définie sur le voisinage 4. Calcul pour chaque niveau de gris n de [0..N-1], d'une mesure d'occupation :

- Occupation(n) = Somme(Somme(Tkl)) + Somme(Somme(Tpq))
  - avec k=[0..n], l=[n+1..N-1]
  - avec p=[n+1..N-1], q=[0..n]

La recherche des minima locaux de occupation(n) se fait sur une plage de *length* niveaux de gris de part et d'autre du niveau de gris n.

Remarque: Cet opérateur ne fonctionne que sur des images de Char parce qu'il faut que les transitions Tkl soient significatives (ie, nombre de (k,l) restreints). Il faut donc s'arranger pour transformer les autres types d'images en image de Uchar.

L'image de sortie *im_out* est contruite avec les seuils détectés, telle que:

`im_out[y][x]=seuil[k] si seuil[k-1]<im_out[y][x]<=seuil[k].`

Le dernier seuil est égal à la valeur maximale 255.

une image de char).

Paramètres

- *length* définit la longueur d'une plage homogène. Plus ce paramètre est grand, moins il y a de maxima régionaux et donc moins il y a de classes en sortie.

Entrées

- *im_in*: une image 2D d'octets (Img2duc, Img3duc).

Sorties

- *im_out*: une image d’octets (Img2duc, Img3duc).

Résultat

Retourne le nombre de seuils détectés.

Exemples

Segmente l’image tangram.pan et affiche le nombre de classes:

```
pweszka 10 tangram.pan out.pan
pstatus
```

Voir aussi

Seuillage

Prototype C++

```
Errc PWeszka( const Img2duc &im_in, Img2duc &im_out, int length );
```

Référence

J.S. Weszka, "Survey of threshold selection techniques", *Computer Graphics and Image Processing*, Vol.7, pp. 259-265, 1978.

Auteur: Régis Clouard

pxor

Ou exclusif binaire entre image ou graphe et différence symétrique entre cartes de régions.

Synopsis

pxor [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

L'opérateur **pxor** effectue le ou exclusif bit à bit entre les deux images d'entrée *im_in1* et *im_in2*.

Pour les images de réelles, le "xor" s'implante avce l'opérateur C '^' et s'applique sur chaque pixel :

```
pixel(im_out) = pixel(im_in1) ^ pixel(im_in2);
```

Pour les images réelles, le "xor" est:

```
if pixel(im1[p]) == pixel(im[2]))
then pixel(imd[p]=0)
else pixel(imd[p]) = pixel(im1[p])+pixel(im2[p])
```

Pour les images couleur et multispectrale, le "xor" est calculé sur chacune des bandes séparément.

Pour les graphes, l'opérateur "xor" s'implante par 'si alors sinon' et s'applique sur les noeuds.

Pour les cartes de régions, le "xor" correspond à la différence symétrique :

```
Union(im_in1,im_in2) - Intersection(im_in1,im_in2).
```

Les deux entrées doivent être de même type.

Entrées

- *im_in1*: une image, un graphe ou une carte de régions.
- *im_in2*: une image, un graphe ou une carte de régions.

Sorties

- *im_out*: un objet du même type que les entrées.

Résultat

Retourne SUCCESS ou FAILURE.
Pour les cartes de régions, retourne la valeur de label maximum.

Exemples

- Sélectionne les pixels qui diffèrent entre deux seuillages de l’image tangram :

```
pbinarization 100 1e30 examples/tangram.pan a.pan
pbinarization 80 1e30 examples/tangram.pan b.pan
pxor a.pan b.pan c.pan
```

Voir aussi

Logique

Prototype C++

```
Errc PXor( const Img2duc &im_in1, const Img2duc &im_in2, Img2duc
&im_out );
```

Auteur: Régis Clouard

pxyz2lab

Changement d'espace couleur de XYZ vers Lab.

Synopsis

pxyz2lab *primaires* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pxyz2lab** permet de changer d'espace couleur en utilisant l'espace Lab.

L'espace couleur LAB définit trois composants:

- L est la luminance,
- a est rouge/bleu
- b est jaune/bleu

La conversion XYZ en Lab utilise la transformation:

```
L=116*((Y/Yn)^(1/3)) si Y/Yn>0.008856
L=903.3*Y/Yn si Y/Yn<=0.008856
a=500*(f(X/Xn)-f(Y/Yn))
b=200*(f(Y/Yn)-f(Z/Zn))

où
f(t)=t^(1/3) si Y/Yn>0.008856
f(t)=7.787*t+16/116 sinon
```

Paramètres

- *primaires* est un entier de l'espace [0..6] qui définit le type de conversion:
  - 0-illuminant E
  - 1-illuminant primaires CIE-DIN
  - 2-illuminant A primaires macbeth colour chart
  - 3-illuminant A primaires CIE
  - 4-illuminant C primaires NTSC
  - 5-illuminant C primaires CIE
  - 6-illuminant D65

Entrées

- *im_in*: les images couleur XYZ.

Sorties

- *im_out*: une image couleur Lab.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pxyz2lab a.pan b.pan
```

Voir aussi

Color

Prototype C++

```
Errc PXYZ2LAB( const Imc2dsf &im_in, Imc2dsf &im_out );
```

Auteur: Olivier Lezoray

pxyz2luv

Changement d'espace couleur de XYZ vers Luv.

Synopsis

pxyz2luv *primaires* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pxyz2luv** permet de passer de l'espace couleur XYZ à l'espace L*u*v. (L: Luminance, u,v : chrominance).

Le rôle de cet espace est de fournir une dérivation perceptuellement uniforme de l'espace CIE XYZ. Perceptuellement uniforme signifie que deux couleurs qui sont à égale distance dans l'espace des couleurs sont à égale distance perceptuellement.

La formule de calcul est donnée par:

```
si (Y/Yn>0.008856)
alors L* = 116*((Y/Yn)^(1/3)-16)
sinon L* = 903.3*Y/Yn

u* = 13*(L*)*(u'-u0')
v* = 13*(L*)*(v'-v0')
```

où u'=4*X/(X+15*Y*+3*Z) et v'=9*Y/(X+15*Y+3*Z) et u0' and v0' ont les mêmes définitions mais appliquées sur le blanc de référence donné par le paramètre *primaire*.

L'image de sortie est une image de float.

Paramètres

- *primaires* est un entier de l'espace [0..6] qui définit le type de conversion:
  - 0-illuminant E
  - 1-illuminant primaires CIE-DIN
  - 2-illuminant A primaires macbeth colour chart
  - 3-illuminant A primaires CIE
  - 4-illuminant C primaires NTSC
  - 5-illuminant C primaires CIE
  - 6-illuminant D65

Entrées

- *im_in*: une image couleur de type XYZ.

Sorties

- *im_out*: une image couleur de type Luv;

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
pxyz2luv 4 a.pan b.pan
```

Voir aussi

Color

Prototype C++

```
Errc PXYZ2LUV( const Imc2dsf &im_in, Imc2dsf &im_out, int primaires
);
```

Auteur: Olivier Lezoray

pxyz2rgb

Changement d'espace couleur de XYZ vers RGB.

Synopsis

pxyz2rgb *primaries* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pxyz2rgb** permet de changer d'espace couleur en utilisant l'espace R,G,B.

L'algorithme de conversion utilise la matrice de transformation pour le cas *primaires* = 4 (illuminant C primaries NTSC):

```
|  1.910  -0.532  -0.288 |
| -0.985   1.999  -0.028 |
|  0.058  -0.118   0.898 | * 255
```

Paramètres

- *primaries* est un entier de l'espace [0..6] qui définit le type de conversion:
  - 0-illuminant E
  - 1-illuminant primaires CIE-DIN
  - 2-illuminant A primaires macbeth colour chart
  - 3-illuminant A primaires CIE
  - 4-illuminant C primaires NTSC
  - 5-illuminant C primaires CIE
  - 6-illuminant D65

Entrées

- *im_in*: les images couleur XYZ.

Sorties

- *im_out*: une image couleur RGB.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertit l’image parrot.pan de rgb en xyz and réciproquement :

```
prgb2xyz 4 parrot.pan a.pan
pxyz2rgb 4 a.pan b.pan
```

Voir aussi

Color

Prototype C++

```
Errc PXYZ2RGB( const Imc2duc &im_in, Imc2dsf &im_out, int primaries
);
```

Auteur: Régis Clouard

pyuv2pan

Conversion d'un fichier de séquence d'image au format YUV (8 bits, codage 4:2:0) en un fichier Pandore.

Synopsis

pyuv2pan width height first_frame last_frame to_rgb *im_in* [*im_out*|-]

Description

L'opérateur **pyuv2pan** permet de transformer un fichier de séquence d'images de format YUV (8 bits, codage 4:2:0) au format Pandore.

Un fichier YUV stocke une séquence d'images couleurs (codée dans l'espace couleur YUV) en données non-compressées sans en-tête. Il est nécessaire de spécifier la taille de l'image lors de l'appel à la fonction de conversion, car il n'y a pas de moyens automatiques de déterminer cette taille. Le fichier Pandore généré est un volume d'images couleurs (Imc3duc), sauf si la séquence initiale ne contient qu'une seule image (Imc2duc). Dans tous les cas, les couleurs de l'image de sortie sont exprimées dans l'espace **RGB**.

Les fichiers .yuv pouvant être particulièrement volumineux, c'est pourquoi il est possible de spécifier un intervalle de frames à récupérer avec les paramètres *first_frame* et *last_frame*.

Paramètres

- *width* spécifie la largeur d'une image de la séquence.
- *height* spécifie la hauteur d'une image de la séquence.
- *first_frame* spécifie le numéro de la première frame de la séquence à récupérer.
- *last_frame* spécifie le numéro de la dernière frame de la séquence à récupérer. Si *last_frame* est égal à -1, **pyuv2pan** lira jusqu'à la dernière frame disponible.
- *to_rgb* spécifie la que l'espace couleur de sortie est RVB si *to_rgb*=1 sinon YCbCr.

Entrées

- *im_in*: un fichier séquence d'images au format YUV (8bits, codage 4:2:0).

Sorties

- *im_out*: une image au format Pandore (Imc2duc ou Imc3duc).

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Convertie la séquence d'image image.yuv en image Pandore 3D avec l'espace couleur RGB:

```
pyuv2pan 256 256 0 -1 1 image.yuv image.pan
```

Voir aussi

Conversion

Prototype C++

```
Errc PYUV2Pan( const char* f_in, Pobject** obj_out, const unsigned
int width, const unsigned int height, const unsigned int
first_frame=0, const int last_frame=-1, const bool to_rgb=false);
```

Avertissement

Auteur: David Tschumperlé

pyuv2rgb

Changement d'espace couleur de YUV vers RGB.

Synopsis

pyuv2rgb [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pyuv2rgb** permet de passer de l'espace couleur Yuv (standard de télévision) à l'espace couleur RGB (Rouge, Vert, Bleu). L'espace couleur Yuv est l'espace couleur adopté pour le format télévision Pal.

La conversion de couleur est un opération linéaire :

```
|  1   0          0.402    |
|  1  -0.344136  -0.714136 |
|  1   0          1.772    |
```

Entrées

- *im_in*: une image couleur Yuv.

Sorties

- *im_out*: les images couleur RGB.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

```
prgb2yuv parrot.pan a.pan
pyuv2rgb a.pan b.pan
```

Voir aussi

prgb2yuv, Color

Prototype C++

```
Errc PYUV2RGB( const Imc2duc &im_in, Imc2dsf &im_out );
```

Auteur: Meftah Boudjelal

pzeboudj

Calcul du critère de qualité basé sur le contraste inter et intra-régions.

Synopsis

pzeboudj [*-m mask*] [*rg_in*|-] [*im_in*|-]

Description

L'opérateur **pzeboudj** calcule un critère de qualité pour l'évaluation d'une segmentation en niveaux de gris tel que défini par R. Zeboudj [*].

La mesure est basée sur le contraste inter et intra-région.
Le contraste d'un pixel s avec son voisin t dans l'image I est mesuré comme suit:

$$c(s,t) = | I(s)- I(t) | / L-1$$

with

L is max(ims)-min(ims).

Le contraste intérieur d'une région Ri est :

$$Ii=1/Ai * sum_{Ri} [max\{c(s,t), t in W(s) inter Ri\}]$$

Le contraste externe d'une région Ri est :

$$Ei=1/li * sum Fi [max\{c(s,t), t in W(s), t not in Ri\}]$$

où Fi est la frontière de la région Ri et li la longueur de Fi.

Le contraste de la région Ri est:

$$C(Ri) = \begin{cases} 1 - Ii/Ei & \text{if } 0 < Ii < Ei; \\ Ei & \text{if } Ii=0; \\ 0 & \text{otherwise;} \end{cases}$$

Finalement, le contraste global est:

$$Contrast = 1/A * sum [Ai.c(Ri)]$$

Le résultat est une valeur dans l'intervalle [0..1]. Plus la valeur du critère de Zeboudj est élevée, meilleure est la segmentation.

Attention: Les régions de label=0 ne sont pas prises en compte pour la mesure.

Entrées

- *rg_in*: une carte de régions.
- *im_in*: une image de niveaux de gris.

Résultat

Retourne un réel positif.
(Utiliser `pstatus` pour récupérer cette valeur).

Exemples

Calcule la mesure de Zeboudj pour une simple segmentation par binarisation :

```
pbinarization 80 1e30 tangram.pan i1.pan
plabeling 8 i1.pan i2.pan
pzeboudj i2.pan tangram.pan
pstatus
```

Voir aussi

Evaluation

Prototype C++

```
Errc PZeboudj( const Reg2d &rg_in, const Img2duc &im_in );
```

Reference

* JP. Cocquerez, S. Philipp, "*Analyse d'images: filtrage et segmentation*", Masson, 1995.

Auteur: Régis Clouard

pzerocross

Localisation des changements de signe des valeurs de pixels.

Synopsis

pzerocross *connexity value* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

L'opérateur **pzerocross** produit une image binaire des points correspondant à un changement de signe des valeurs de pixels de l'*im_in*. La valeur *value* est considérée comme la valeur de changement de signe. Il y a changement de signe, lorsque la valeur d'un pixel:

- est supérieur à *value* et qu'au moins un de ses voisins est inférieur à *value*,
- est inférieur à *value* et que l'un de ses voisins est supérieur à *value*.

L'image de sortie *im_out* est une image binaire construite avec les points de changement de signe à 255. L'image de sortie est forcément de type Uchar (Img2duc ou Img3duc).

Cet opérateur est notamment utilisé pour détecter le passage par 0 du laplacien (Voir Laplacien). Par exemple, avec une image de Uchar, la valeur de coupe est à 127. Pour une image Slong, la valeur de coupe est à 0.

Paramètres

- *connexity* définit la notion de voisinage (4, 8 en 2D et 6, 26 en 3D). Ce paramètre est ignoré pour les graphes.
- *value* appartient à l'intervalle de niveau de gris accepté par l'image *im_in*.

Entrées

- *im_in*: une image de niveaux de gris.

Sorties

- *im_out*: une image binaire.

Résultat

Retourne SUCCESS ou FAILURE.

Exemples

Détection de contours à partir de l'algorithme DOG (Difference de Gaussiennes):

```
pexponentialfiltering 0.2 tangram.pan a.pan
pexponentialfiltering 0.8 tangram.pan b.pan
psub a.pan b.pan c.pan
pzerocross 8 0 c.pan out.pan
```

Voir aussi

Détection contours

Prototype C++

```
Errc PZeroCross( const Img2duc &im_in, Img2duc &im_out, int
connexity, Uchar value );
```

Auteur: Régis Clouard