Operator Index

File Format Conversion

Converts between image file format (gif, bmp, tiff, pandore, etc).

- pbmp2pan - Converts BMP image file to Pandore image file.
- ppan2bmp - Converts Pandore file to BMP image file.
- pgif2pan - Converts GIF image file to Pandore image file.
- ppan2gif - Converts Pandore image file to GIF image file.
- ppan2ppm - Converts Pandore image file to PPM (ascii) image file.
- pppm2pan - Converts PPM, PGM, or PBM image file to Pandore file.
- ptiff2pan - Converts TIFF image file to Pandore image file.
- ppan2tiff - Converts Pandore image file to TIFF image file.
- pvff2pan - Converts VFF image file to Pandore image file.
- ppan2vff - Converts Pandore image file to VFF image file (Sunvision).
- ppan2pan - Converts any Pandore file format to current Pandore file format.
- pras2pan - Converts Sun raster image file to Pandore image file.
- praw2pan - Converts raw image file to Pandore image file.
- ppan2raw - Converts Pandore file to raw file.
- ppan2ps - Converts Pandore image file to Encapsulted PostScript file.
- ppan2txt - Converts Pandore image file to text file.
- ptxt2pan - Converts text file to Pandore image file.
- pyuv2pan - Converts a YUV image sequence file to a Pandore image file.
- pfits2pan - Converts FITS (Flexible Image Transport System) image file to Pandore image file.
- ppan2fits - Converts Pandore gray-scale image (1D, 2D or 3D) to FITS (Flexible Image Transport System) image file.
- pparrec2pan - Converts PAR/REC format image file (Philips Medical System) to Pandore image file.
- panalyze2pan - Converts Pandore image file to ANALYZE 7.5 image file.
- ppan2analyze - Converts ANALYZE 7.5 image file to Pandore image file.
- ppan2d23d - Converts a series of Pandore 2D image files to a unique 3D image file.
- ppan3d22d - Converts Pandore 3D image file to a series of 2D Pandore image files.

Pandore Type Conversion

Casts between Pandore object format: Float image to Uchar image, Image to region map, etc.

- parray2im - Converts pixel array to image.
- pim2array - Converts image to pixel array.
- prg2im - Converts region map to signed long image.
- pim2rg - Converts grayscale image to region map.
- pgr2im - Converts graph to float image.

- prg2gr - Converts region map to graph.
- pgr2rg - Converts graph to region map.
- pim2sf - Converts any image type to float image.
- pim2uc - Converts any image type to unsigned char image.
- pim2sl - Converts any image type to signed long image.
- pim2d23d - Converts 2D image or region map to 3D image or region map with 1 slice.
- pim3d22d - Converts 3D image or region map with one slice to 2D image or region map.
- pimc2img - Converts color image to grayscale image.
- pimc2imx - Creates a multispectral image with a color image.
- pimg2imc - Creates a color image from 3 grayscale images.
- pimg2imx - Creates a multispectral image from several grayscale images.
- pimx2img - Converts one band of multispectral image to grayscale image.
- pimx2imc - Converts multispectral image to color image.

Color Space Conversion

Converts color spaces.

- phsi2rgb - Converts HSI color image to RGB color image.
- phsl2rgb - Converts HSL color image to RGB color image.
- plab2lch - Converts Lab color image to LCH color image.
- pluv2lch - Converts L*u*v color image to LCH color image.
- prgb2pca - Converts rgb color image to principal components.
- prgb2ast - Converts rgb color image to AST color image.
- prgb2gray - Converts RGB color image to gray space image.
- prgb2hsi - Converts RGB color image to HSI color image.
- prgb2hsl - Converts RGB color image to HSL color image.
- prgb2i1i2i3 - Converts RGB color image to (i1,i2,i3) color image.
- prgb2wry - Converts RGB color space to (wb,rg,yb) color space.
- prgb2xyz - Converts RGB color image to XYZ color image.
- prgb2rgnbn - Converts RGB color image to normalized RGB color image.
- prgb2ycbcr - Converts RGB color image to YCbCr color image.
- prgb2ych1ch2 - Converts RGB color space to YCh1Ch2 color image.
- prgb2yiq - Converts RGB color image to YIQ color image.
- prgb2yuv - Converts RGB color image to YUV color image.
- pyuv2rgb - Converts YUV color image to RGB color image.
- pxyz2lab - Converts XYZ color image to Lab color image.
- pxyz2luv - Converts XYZ color image to L*u*v color image.
- pxyz2rgb - Converts XYZ image to RGB image.
- pgray2bw - Converts gray scale image to black and white image.
- pgray2falsecolor - Converts gray level image to false color image.

Arithmetic

Performs binary and unary operations on image, graph and region map.

- padd - Performs addition between images or graphs.
- pdif - Performs difference between images or graphs and non symmetrical difference between region maps.
- psub - Performs subtraction between images or graphs and non symmetrical difference between region maps.
- pmult - Performs multiplication between images or graphs.
- pdiv - Performs division between images or graphs.
- pmean - Performs average between images or graphs.
- pblend - Performs alpha blending of image or graph.
- ppow - Computes nth power of an image or a graph.
- psqrt - Computes square root of image or graph.
- pexp - Computes exponential of image or graph.
- plog - Computes natural logarithm of image or graph.
- pmax - Performs maximum values between image or a graph.
- pmin - Computes minimum values between image or graph.
- pabs - Computes absolute value of image or graph.
- pround - Computes round integral value of image or graph.
- pclipvalues - Clips pixel values inside the specified range.
- pnormalization - Performs normalization of image or graph.
- pconvolution - Convolves image with kernel.
- psetcst - Sets constant to image, graph or region map.
- pdivcst - Divides constant to image, graph or region map.
- pmultcst - Multiplies constant to image, graph or region map.
- paddcst - Adds constant to image, graph or region map.
- paddval - Adds constants stored in collection to image bands.
- psubval - Subtracts constants stored in collection to image bands.
- pmultval - Multiplies image bands with constants stored in collection.
- pdivval - Divides image bands with constants stored in collection.
- plipadd - Performs image addition according to the LIP model.
- plipsub - Performs image subtraction according to the LIP model.
- plipmultcst - Performs scalar multiplication of image according to the LIP model.

Logic

Performs binary and logical operation between image and graph and set operation between region map.

- pand - Performs binary and between images or graphs and intersection between region maps.
- por - Performs binary or between images or graphs and union between region maps.
- pxor - Performs binary xor between images or graphs and symmetrical difference between region maps.
- pinverse - Performs binary inversion of image or graph and inversion of region map labels.
- pnot - Performs logical negation of image or graph and complementary of region map.
- pmask - Performs masking of image, graph or region map by an image or a region map.

Geometrical Transformation

Performs basic geometrical transformation of image, graph or region map contents.

- paddborder - Performs resizing of image or region map by adding a border.
- pmaxprojection - Performs mean value orthogonal projection along main axis.
- pmeanprojection - Performs mean value orthogonal projection along main axis.
- pflip - Performs flip transformation of image or region map.
- protation - Performs rotation of image or region map.
- ptranslation - Performs translation of image or region map.
- pscrolling - Performs a rolling up translation of image or region map.
- ptransposition - Performs a transposition of image.
- presize - Performs an affine resizing of image or region map.
- prescale - Performs an affine rescaling of image, region map or graph using the nearest neighbor interpolation.
- plinearrescale - Performs an affine rescaling of image using the linear interpolation.
- pbicubicrescale - Performs an affine rescaling of 2D image using the bicubic interpolation.
- pbellcrescale - Performs a rescaling of image using the Bell algorithm.
- phermiterescale - Performs a rescaling of image using the Hermite algorithm.
- planczosrescale - Performs a rescaling of image using the Lanczos algorithm.
- pmitchellrescale - Performs a rescaling of image using the Mitchell algorithm.
- pquadraticbsplinerescale - Performs a rescaling of image using the Quadratic Bezier Spline interpolation.
- ptrianglerescale - Performs a rescaling of image using triangular filter.

Image Utility

Performs basic image transformations.

- paddnoise - Adds artificial noise to image.
- pcliparea - Select rectangular region area of image, region map or graph.
- psetborder - Sets image border to specified value.
- pextractsubimage - Extracts subimage from image.
- pinsertsubimage - Inserts subimage into image.
- pnewimage - Creates a new image from dimensions.
- pshapedesign - Creates a new image with synthetic shape.
- pmergeimages - Merges 4 images into one image.
- psplitimage - Splits image into 4 subimages.
- pgraylevel2depth - Converts gray levels to depth values.
- pdepth2graylevel - Converts depth values to gray levels.
- paddslice - Adds slice to 3D image.
- premoveslice - Removes slice to 3D image.
- pgetband - Gets a band in a multispectral image.
- pgetslice - Gets slice from 3D image as 2D image.
- psetlice - Sets slice to 3D image.

Lut Transformation

Performs look-up table transformation to enhance the image lisiblity.

- psharp - Performs contrast sharpening using laplacian unsharp masking.
- pextremumsharpening - Performs constrast sharpening using extremum values.
- plineartransform - Performs linear transform of the gray-levels.
- plogtransform - Performs logarithmic and exponential transforms of the gray-levels.
- ppowerlawtransform - Performs power-law transform of the gray-levels.
- phistogramequalization - Performs contrast stretching using histogram equalization.
- phistogramspecification - Performs contrast stretching using histogram specification.

Spatial Filtering

Performs some spatial linear or non linear filtering.

- pvariancefiltering - Performs variance filtering on image.
- pmedianfiltering - Performs median filtering on image.
- pmeanfiltering - Performs mean filtering on image or graph.
- pgaussianfiltering - Performs gaussian filtering on image.
- pexponentialfiltering - Performs exponential filtering on image.
- padaptivemeanfiltering - Performs adaptive mean filtering on image.
- pderichesmoothing - Performs Deriche filtering on image.
- pshensmoothing - Performs Shen-Castan smoothing on image.
- pmalikperonafiltering - Performs non linear diffusion smoothing.
- pmcmfiltering - Performs Mean Curvature Motion filtering on image.
- pnagaofiltering - Performs Nagao filtering on image.
- poutrangefiltering - Performs outrange filtering on image.
- psigmafiltering - Performs sigma filtering on image.
- psnnfiltering - Performs Symmetric Nearest Neighborhood filtering on image.
- ppeergroupfiltering - Performs Peer Group filtering on color image.
- pdenoisePDE - Performs anisotropic smoothing on image.

Surface Fitting

Approximates image content to flat background.

- plinearregression - Approximates image to flat background using linear regression.
- ppolynomialfitting - Approximates image to flat background using polynomial fitting.
- plegendrepolynomialfitting - Approximates image to flat background using Legendre polynomial fitting.

Frequency Domain

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 - Performs Fast Fourier Transform.
- pifft - Performs inverse Fast Fourier Transform.
- pbutterworthfilter - Designs lowpass, highpass, bandpass or bandreject Butterworth filter.
- pgaussianfilter - Designs lowpass, highpass, bandpass or bandreject Gaussian filter.
- pmodulus - Computes modulus between 2 images.
- pphase - Computes phase between 2 images.
- pfftconvolution - Performs convolution of image by kernel.
- pfftdeconvolution - Performs deconvolution of image by kernel.
- pfftcorrelation - Performs correlation between two images.
- pfftshift - Shifts images in FFT image.
- pqmf - Designs Quadratic Mirror Filter for wavelet transform.
- pdwt - Performs Direct Wavelet Transform.
- pidwt - Performs inverse Direct Wavelet Transform.
- psetsubband - Gets one subband image from DWT image.
- pgetsubband - Sets subband into DWT image.

Mathematical Morphology Domain

Performs some morphological transformations based on set operations such as erosion and dilatation from a structuring element.

- pdilatation - Performs morphological dilatation.
- perosion - Perfoms morphological erosion.
- psedesign - Designs structuring element as image.
- psedilatation - Performs morphological dilatation.
- pseerosion - Performs morphological erosion.
- plineardilatation - Performs linear dilatation.
- plinearerosion - Performs linear erosion.
- pgeodesicdilatation - Performs geodesic dilatation.
- pgeodesicerosion - Performs geodesic erosion.
- pareaopening - Performs area opening (minima killer).
- pareaclosing - Performs area closing (maxima killer).
- pwatershed - Performs the watershed on image.
- pdilatationreconstruction - Performs reconstruction by dilatation.
- perosionreconstruction - Performs reconstruction by erosion.
- pskeletonization - Computes the skeleton of 2D objects.
- phomotopicskeletonization - Computes the homotopic skeleton of 3D objects.

Point of Interest Detection

Detects points of interest such as corners or junctions.

- pharris - Performs Harris corner detection.
- psusan - Performs Susan corner detection.

Edge Detection

Detects and localizes edges.

- pgradneumann - Computes the discrete gradient by forward finite differences and Neumann boundary conditions.
- pdivneumann - Computes the divergence by backward finite differences.
- pgradient - Computes the gradient magnitude and direction.
- plaplacian - Computes the Laplacian magnitude.
- pprewitt - Computes the Prewitt gradient magnitude.
- proberts - Computes the Roberts gradient magnitude.
- psobel - Computes the Sobel gradient magnitude.
- pderiche - Computes gradient magnitude and maxima localization using Deriche algorithm.
- pshen - Computes gradient magnitude and maxima localization using Shen-Castan's algorithm.
- pgradientthreshold - Estimates the noise level in a gradient image.
- pnonmaximasuppression - Performs non-maxima suppression for edge detection.
- pzerocross - Locates zero crossing.

Contour Processing

Contours are chains of connected non null pixels relying on a null background. There are connected according to 8-connexity in 2D and the 26-connexity in 3D. Contour image are grayscale images.

- pdistance - Computes euclidean distance map to nearest contours.
- pdistance1 - Computes distance map to nearest contour.
- pcontourextensionrect - Performs rectangular-shaped extension of end points.
- pcontourextensionconic - Performs cone-shaped extension of end points.
- pedgeclosing - Performs edge closing from gradient.
- phoughlines - Detects straight lines from a set of contours.
- ppostthinning - Performs contour postthinning to guaranty 8-connexity (or 26-connexity).
- pellipsoidalapproximation - Performs ellipsoidal approximation of closed contour.
- ppolygonalapproximation - Performs polygonal approximation of contours.
- pbarbremoval - Removes barbs from length.
- pcontourselection - Selects contour chain from length.
- pclosedcontourselection - Selects closed contour from length.
- popencontourselection - Selects open contour from length.

Thresholding

Segments image using pixel classification according to their values.

- pbinarization - Performs binary thresholding on image and graph.
- padaptivebinarization - Performs binarization on image using a local adaptive thresholding.
- pcorrelationbinarization - Performs binarization on image using maximum correlation criterion.
- pentropybinarization.html - Performs binarization on image using maximum entropy criterion.
- pvariancebinarization - Performs binarization on image using interclass variance criterion.
- pthresholding - Performs thresholding on image, region map or graph.

- pcontrastthresholding - Performs multi-thresholding on image based on the boundary contrast value.
- phistothresholding - Performs multi-thresholding on image using histogram analysis.
- pmassthresholding - Performs thresholding on image using gray level percent.
- pfuzzyclustering - Performs pixel classification using fuzzy k-means algorithm.
- pentropythresholding - Performs multi-thresholding on image based on the entropy value.
- pchanda - Performs multi-thresholding on image using Chanda algorithm.
- pderavi - Performs multi-thresholding on image using Deravi algorithm.
- pfisher - Performs multi-thresholding on image using Fisher algorithm.
- pweszka - Performs multi-thresholding on image using Weszka algorithm.

Segmentation

- pboundarylabeling - Performs region labeling from boundary.
- plabeling - Performs region labeling.
- pcontrastquadtree - Performs quadtree (or octree) segmentation based on contrast uniformity.
- pcontrast1quadtree - Performs quadtree (or octree) segmentation based on contrast criterion.
- pentropyquadtree - Performs quadtree segmentation based on entropy uniformity.
- puniformityquadtree - Performs quadtree (or octree) segmentation based on uniformity degree.
- pvariancequadtree - Performs quadtree (or octree) segmentation based on variance uniformity.
- pcontrastaggregation - Performs pixel aggregation based on contrast criterion.
- pedgebasedragpruning - Cut adjacency link between regions that are separated by an edge.
- pmeanaggregation - Performs pixel aggregation based on mean criterion.
- pgaussaggregation - Performs pixel aggregation based on gaussian criterion.
- pvarianceaggregation - Performs pixel aggregation based on variance criterion.
- pcontrastmerging - Performs priority region merging based on contrast criterion.
- pentropymerging - Performs priority region merging based on entropy criterion.
- pmeanshiftsegmentation Performs pixel classification on image using mean shift algorithm.
- pmeanmerging - Performs priority region merging based on mean criterion.
- pmumfordshahmerging - Performs priority region merging based on Mumford-Shah criterion.
- puniformitymerging - Performs priority region merging based on uniformity criterion.
- pvariancemerging - Performs priority region merging based on variance criterion.
- pboundarymerging - Performs priority region merging based on uniformity criterion.
- phistomerging - Perfoms priority region merging based on histogram correlation.
- plabelmerging - Merges two regions from label value.
- pinnermerging - Performs inner region merging.
- pinnermostmerging - Performs inner-most region merging.
- pvoronoi - Calculates the Voronoi diagram.

Region Processing

Performs operations on region map such as region selection and relabeling. Inputs and outputs are region maps.

- pfillhole - Fills region holes.
- pconvexhull - Builds convex hull of regions.
- pboundary - Locates region boundary.

- pboundingbox - Builds the bounding box around regions.
- pcenterofmass - Locates the centre of mass of regions.
- plabelselection - Selects region by label value.
- pinnerselection - Selects inner regions.
- poutborderselection - Selects regions that do not touch the border.
- penergyselection - Selects regions from energy value.
- pmaximumselection - Selects regions from maximum grayscale value.
- pminimumselection - Selects regions from minimum grayscale value.
- pmeanselection - Selects regions from mean grayscale value.
- pvarianceselection - Selects regions from variance factor.
- pcompactnessselection - Selects regions from compactness factor.
- pconvexityselection - Selects regions from convexity degree.
- pelongationselection - Selects regions from elongation factor.
- peulernumberselection - Selects regions from Euler number.
- peccentricityselection - Selects regions from eccentricity degree.
- porientationselection - Selects regions from orientation degree.
- pperimeterselection - Selects regions from perimeter length.
- prectangularityselection - Selects regions from rectangularity degree.
- psphericityselection - Selects regions from sphericity degree.
- psizeselection - Selects regions from size.
- pareaselection - Selects regions from area size.
- pvolumeselection - Selects regions from volume factor.
- prelabelingfromarray - Relabels regions from label array.
- prelabelingwithgraph - Relabels region and related graph node.

Region Features Extraction

Measures some region topological, geometrical and photometrical features.

- pareadisorderfactor - Measures area disorder factor.
- pregioncompactness - Measures region compactness factor.
- pregionconvexity -Measures region convexity degree.
- pregionelongation - Measure region elongation factor.
- pregionenergy -Measures region energy value.
- pregioneulernumber - Measures region Euler number.
- pregioneccentricity - Measures region eccentricity degree.
- pregionmaximum - Measures region maximum grayscale value.
- pregionminimum - Measures region minimum grayscale value.
- pregionmean -Measures region mean grayscale value.
- pregionorientation -Measures region orientation degree.
- pregionperimeter - Measures region perimeter length.
- pregionrectangularity - Measures region rectangularity degree.
- pregionsphericity - Measures region sphericity degree.
- pregionvariance - Measures region variance.
- pregionarea - Measures region area size.
- pregionvolume - Measures region volume.

Image Features Extraction

Extraction de caractéristiques statistique sur ou entre images.

- plocalextrema - Computes local extremum values of grayscale image.
- plocalmaxima - Computes local maximum values of grayscale image or graph.
- plocalminima - Computes local minimum values of grayscale image or graph.
- pcontrastvalue - Measures the global contrast value of grayscale image or graph.
- pcontrast1value - Measures the global contrast value of grayscale image or graph.
- penergyvalue - Measures global energy of grayscale image or graph.
- pentropyvalue - Measures global entropy of grayscale image or graph.
- psumvalue - Measures global sum of grayscale image or graph.
- pvariancevalue - Measures global variance of grayscale image or graph.
- pmaximumvalue - Measures global maximum value of image, region map or graph.
- pminimumvalue - Measures global minimum value of image, region map or graph.
- pmeanvalue - Measures global mean value of grayscale image or graph.
- pmedianvalue - Measures median value of grayscale image.
- ppixelvalue - Gets pixel value from grayscale image and region map.
- pvaluenumber - Returns the amount of non null pixels of image, region map or graph.
- pvalueclassnumber - Counts the number of different used values of image, region map or graph.
- pvaluerank - Gets the nth value for image, region map or graph.
- phistogram - Creates histogram from grayscale image.

Evaluation

Computes discrepancy and goodness measures to evaluate the quality of image processing.

- pvinet - Computes the discrepancy measure between two region maps based on the number of mis-segmented pixels.
- pborsotti - Computes the goodness measure based on the number, area and variance of regions.
- pzeboudj - Computes the goodness measure based on inter and intra-region contrast.
- pinterregioncontrast.html - Computes the goodness measure based on inter-region contrast.
- pintraregionuniformity.html - Computes the goodness measure based on intra-region uniformity.
- pmse - Computes the Mean Square Error.
- ppsnr - Computes the Peak Signal to Noise Ratio.
- psnr - Computes the Signal-to-Noise Ratio.

Motion

Recontructs the motion.

- pblockmatching - Motion estimation between 2 images using block matching.
- pplotquiver - Renders vector field from multispectral images with 2 bands.
- pregistrationPDE - Computes the displacement field between two input images.

Array

Performs some arithmetic operations on array stored in collection.

- pcreatearray - Creates array in collection.
- parray2array - Coverts array type.
- parraygetvalue - Gets value from array.
- parraysize - Returns the size of array in a collection.
- parrayargmax - Calculates the maximum values between arrays.
- parraycovarmat - Calculates covariance matrix of arrays items.
- parraymean - Calculates mean of array items.
- parrayeuclideannorm - Calculates euclidean norm of arrays items.
- parraynorm - Performs normalization between 0 and 1 of items of one array.
- parraysnorm - Performs normalization between 0 and 1 of items of several arrays.

Graph

Performs somne operations on graph. A graph is a set of vertices (nodes) linked by weighted edges. A vertex (node) references an item (region, point, ...) in an array.

- pbetagraph - Builds the beta graph.
- psig - Builds the Sphere of Influence Graph.
- pmst - Builds the Minimum Spanning Tree of graph.
- pdelaunay - Builds the Delaunay graph from region map.
- pgraphpruning - Performs graph pruning.
- pedgecutting - Perform edge cuttin.
- pleafcutting - Performs leaf cutting.
- pgraphneighbours - Valuates nodes with the number of neighbours.
- pgraphvisu - Visualizes node and edge of graph.
- pedgevisu - Visualization of graph edge weights.
- pnodevisu - Visualizes graph node values.
- pnodedisc - Visualizes graph node values.

Collection

Manages collection contents. A collection is a bundle of values and Pandore objects identified by a name.

- pcolcatenateitem - Catenates two collections.
- pcolremoveitem - Removes item from collection.
- pcolrenameitem - Renames item in collection.
- pcolgetvalue - Gets numerical value from a collection.
- pcolsetvalue - Sets numerical value to collection.
- pcolgetimages - Extracts images from collection.
- pcolgetobject - Gets Pandore object from collection.
- pcolsetobject - Sets Pandore object to collection.
- pobject2col - Creates collection from Pandore object.

- pcol2txt - Converts collection to text file.
- ptxt2col - Builds a collection from text file.

Objects Classification

Performs sone object classification. Object can be anything like pixel or region.

- pgaussclassification - Performs gauss clustering on a set of objects.
- pkmeans - Performs K-means clustering on a set of objects.
- pknn - Performs K-nearest neighbors clustering on a set of objects.

Visualization

Displays graphically or textually Pandore object contents.

- pvisu - Displays Pandore object.
- pdraw - Draws by hand onto Pandore object.
- pcontentsdisplay - Displays the contents of Pandore object.
- pplot1d - Renders a plot of a 1D function as a color image.
- pcolorcube - Creates the RGB cube of image color repartition.
- pcolorize - Colorization of regions with internal average value.
- psuperimposition - Mask superimposition onto image.

Information

Displays information about Pandore objects.

- pproperty - Gets Pandore object attribute value.
- pfile - Displays properties of Pandore file.
- psetstatus - Sets the current status value.
- pstatus - Prints out the value returned by the last executed operator.
- pman - Finds and displays reference manual pages.
- pversion - Prints the current Pandore version number.

Miscellaneous

Some miscellaneous image operations not in direct relationship with image processing.

- prds - Builds Random Dot Stereogram.
- pstereogram - Builds Autostereogram.
- psubsampling - Sub-sampling of an image.

pabs

Computes absolute value of image or graph.

Synopsis

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

Description

pabs computes the absolute value of the input *im_in*.

If *im_in* is an image then the new image *im_out* is built with the absolute value of each pixel:

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

For non signed image, *im_out* is just a copy of *im_in*.

For color or multispectral image, the absolute value is computed separately on each band.

If *im_in* is a graph then the new graph *im_out* is built with the absolute value of each node value.

The output *im_in* type is the same type as the input *im_in* type.

Inputs

- *im_in*: an image or a graph.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Computes the difference between images a.pan and b.pan and stores the result in image d.pan:

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

See also

Arithmetic

C++ prototype

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

Version française

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

Author: Régis Clouard

padaptivebinarization

Performs binarization on image using a local adaptive thresholding.

Synopsis

padaptivebinarization *halfsize* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

padaptivebinarization classifies pixels of the input image *im_in* into 2 clusters. The binarization uses a local adaptive binarization based on examination the mean values of the local neighborhood of each pixel. The neighbour is defined as the *halfsize * halfsize* pixels around the central pixel.

The algorithm is as follows for each pixel p:

```
if im_in[p] > mean(neighbour value)
then im_out[p]=255;
else im_out[p]=0;
```

Parameters

- *halfsize_in*: the half size of the neighbourhood. It means that halfsize*halfsize pixels around the central pixel are used to compute the locale mean.

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).

Outputs

- *im_out*: a binary image of bytes (Img2duc, Img3duc).

Result

Returns SUCCESS or FAILURE.

Examples

Segments the tangram pieces:

```
padativebinarization 7 tangram.pan a.pan
```

See also

Thresholding

C++ prototype

```
Errc PadativeBinarization( const Img2duc &im_in, Img2duc &im_out,
int halfsize );
```

Version française

Binarisation de l’image adaptation locale.

Author: Régis Clouard

padaptivemeanfiltering

Performs adaptive mean filtering on image.

Synopsis

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

Description

padaptivemeanfiltering applies an adaptive mean filter to the input image *im_in*. Each pixel is replaced by the mean of the neighbors of its neighbors that have the minimum gradient magnitude. *im_in2* contains the gradient magnitude for each pixel.

The image border (of size 2) is not considered for processing. The output image border is just a copy of the input image border.

Parameters

- *connexity* specifies the neighborhood relation (4 or 8 in 2D).

Inputs

- *im_in1*: a 2D image.
- *im_in2*: a 2D image of gradient magnitude.

Outputs

- *im_out*: an image of the same type as the input image *im_in1*.

Result

Returns SUCCESS or FAILURE.

Examples

Applies an adaptive mean filter to tangram.pan:

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

See also

Filtering

C++ prototype

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

Version française

Lissage d’une image préservant les contours.

Author: Régis Clouard

padd

Performs addition between images or graphs.

Synopsis

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

Description

padd computes the addition of the two inputs *im_in1* and *im_in2*.

If *im_in1* and *im_in2* are images then the new image *im_out* is built with the addition of each pixel:

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

The two inputs must be of the same type.

The output type *im_out* depends on input type:

- Long if inputs are Uchar images.
- Long if inputs are Long images.
- Float if inputs are Float images.

For color or multispectral image, the addition is computed separately on each band.

If *im_in1* and *im_in2* are graphs then the new graph *im_out* is built with the addition of each node values.

Inputs

- *im_in1*: an image or a graph.
- *im_in2*: an image or a graph.

Outputs

- *im_out*: an image or a graph. The type depends on input type:
    - Long if inputs are Uchar images.
    - Long if inputs are Long images.
    - Float if inputs are Float images.

Result

Returns SUCCESS or FAILURE.

Examples

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

See also

Arithmetic

C++ prototype

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

Version française

Addition de 2 images ou de 2 graphes.

Author: Régis Clouard

paddborder

Performs resizing of image or region map by adding an empty border.

Synopsis

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

Description

paddborder magnifies the input image *im_in* by adding a border around. The size of the border is defined by *ll* + *lr* width, *hu* + *hd* height and *df* + *db* depth.

Pixels in the new border is set to 0.

Parameters

- *df* is the size of the forward border (3D image).
- *db* is the size of the backward border (3D image).
- *hu* is the size of the higher border.
- *hd* is the size of the lower border.
- *ll* is the size of left border.
- *lr* is the size of right border.

  df and *db* are ignored in case of 2D image.

Inputs

- *im_in*: an image or a region map.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Adds an empty border around tangram.pan

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

See also

Transformation

C++ prototype

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

Version française

Agrandissement d’une image en lui ajoutant un bord à 0.

Author: Régis Clouard

paddcst

Adds constant to image, graph or region map.

Synopsis

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

Description

paddcst builds the new output *im_out* by adding the specified constant to each value of *im_in*.

If *im_in* is an image then **paddcst** adds the specified value to each pixel. The values are clipped if they are greater than the maximum allowed value or lower than the minimum:

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

For color or multispectral image, **padddcst** is computed separately on each band.

For region map, **paddcst** adds the specified value to each label.

For graph, **paddcst** adds the specified value to each node value.

The output file is of the same type as the input file.

Parameters

- *cst* is a real value.

Inputs

- *im_in*: an image, a graph or a region map.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

For region map, returns the new higher label value.

Examples

Adds 10 to each tangram.pan pixel values:

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

See also

Arithmetic

C++ prototype

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

Version française

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

Author: Régis Clouard

paddnoise

Adds artificial noise to image.

Synopsis

paddnoise *law mean std_dev* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

paddnoise adds artificial noise to the input image *im_in*. Several distribution laws exist either additive or multiplicative and can be chosen from the parameter *law*. The output image *im_out* is built as follows:

- additive law: im_out = im_in + im_noise;
- multiplicative law: im_out = im_in * im_noise;

where `im_noise` is a noise image built from the specified distribution law.

Let u1 and u2 be two random variables uniformly distributed on the space [0..1]. The noise value for pixel p is computed as follows:

- gaussian law (Box Muller algorithm):

```
z0=sqrt(-2.0*log(u1))*cos(2.0*M_PI*u2);
im_noise[p] = std_dev*z0 + mean;
```

- exponential law (inversion method):

```
z0=-1.0*log(u1);
im_noise[p]= std_dev*z0 + mean;
```

- uniform law (rem : std_dev = (max-min)/sqrt(12)):

```
z0=(u1-0.5)*sqrt(12.0);
im_noise[p] = std_dev*z0 + mean;
```

- triangular law:

```
z0=(u1+u2-1.0)*sqrt(6.0);
im_noise[p] = std_dev*z0 + mean;
```

Parameters

- *law* is an integer which specifies the distribution law:
  - 1: additive gaussian noise.
  - 2: additive exponential noise.
  - 3: additive uniform noise.

    - 4: additive triangular noise.
    - 11: multiplicative gaussian noise.
    - 12: multiplicative exponential noise.
    - 13: multiplicative uniform noise.
    - 14: multiplicative triangular noise.
- *mean* is a real which defines the mean of the distribution.
- *std_dev* is a real which defines the standard deviation of the distribution.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE in case of bad parameter values.

Examples

Adds gaussian noise with mean 0 and standard deviation 1.5 to tangram.pan image and then computes the PSNR for the meanfilter smoothing operator:

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

See also

Utility

C++ prototype

```
Errc PAddNoise( const Img2duc &im_in, Img2duc &im_out, int law,
Float mean, Float std_dev );
```

Version française

Génération de bruit aléatoire sur une image.

Author: Régis Clouard

paddslice

Adds slice to 3D image.

Synopsis

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

Description

paddslice adds a 2D image at the end or at the beginning of a 3D image. The new image *im_out* has one slice more than the input image *im_in1*.

The first 3D image can be built with operator **pim2d23d**.

The 2D image is added at the beginning if parameter *direction* is negative or at the end if it is positive.

The result image *im_out* is of the same type as the two input images.

Parameters

- *direction* specifies whether the 2D image is added at the beginning (*direction* < 0) or at the end (*direction* > 0) of the 3D image.

Inputs

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

Outputs

- *im_out*: a 3D image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Adds a2d.pan to the end of the a3d.pan:

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

See also

Utility, pinsertslice, premoveslice, pim2d23d

C++ prototype

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

Version française

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

Author: Régis Clouard

paddval

Adds constants stored in collection to image bands.

Synopsis

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

Description

paddval builds the new output *im_out* by adding constants stored in the collection *col_in* to each band of the input image *im_in*. The first constant in the collection is added to the first band, the second constant to the second, etc.

The values are clipped if they are greater than the maximum allowed value or lower than the minimum:

```
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;
```

The output file is of the same type as the input file.

Inputs

- *col_in*: a collection with a number of float values equals to the number of bands of the input image.
- *im_in*: an image.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Adds the mean value of tangram.pan image to tangram.pan:

```
pmeanvalue tangram.pan col.pan
paddval col.pan tangram.pan a.pan
```

More examples

See also

Arithmetic

C++ prototype

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

Version française

Addition d’une image avec des constantes stockÃ©es dans une collection.

Author: Régis Clouard

panalyze2pan

Converts ANALYZE 7.5 image file to Pandore image file.

Synopsis

panalyze2pan *im_in* [*im_out*|-]

Description

panalyze2pan converts an ANALYZE 7.5 image file to a Pandore image file.

An ANALYZE (7.5) format image consists of two files in the same directory and with the same base name:

- a header file (suffixed .hdr). It contains information about the .img file, such as the volume represented by each number in the image (voxel size) and the number of pixels in the X, Y and Z directions. This header contains fields of text, floating point, integer and other information.
- an image file (suffixed .img). It contains the numbers that make up the information in the image.

The input file *im_in* is one of the two ANALYZE files. The second file is then read from the same directory and with the same base name.

The result image *im_out* is a 3D multispectral image of floats (Imx3duc).

Inputs

- *im_in*: an ANALYZE 7.5 file name (either .hdr .img).

Inputs

- *im_out*: a Pandore 3D image of Float (Imx3dsf).

Result

Returns SUCCESS or FAILURE.

Examples

Converts image "brain" to Pandore image "a.pan" and then displays the band #0.

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

See also

Conversion, ppan2analyze

C++ prototype

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

Version française

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

Important notice

Author: David Tschumperlé

pand

Performs binary and between images or graphs and intersection between region maps.

Synopsis

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

Description

pand performs a bitwise "and" between values of the two inputs *im_in1* and *im_in2*.

For integer images, the "and" operator uses the ’&’ C operator and is applied on each pixel:

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

For real image, the "and" operator uses the ’*’ C operator:

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

For color or multispectral images, the "and" operator is computed separately on each band.

For graphs, the "and" operator is "*" and it is applied on each node values.

For region maps, the "and" operator is the intersection between regions. The result *im_out* is a new region map with common regions, giving preference to smaller regions. Common regions are regions at the same location and a label > 0. It is not necessary that the regions have the same label value.

The two inputs must be of the same type.

Inputs

- *im_in1*: an image, a graph or a region map.
- *im_in2*: an image, a graph or a region map.

Outputs

- *im_out*: an object of the same type as im_in1 and im_in2.

Result

Returns SUCCESS or FAILURE.

For region map, returns the new higher label value.

Examples

```
pand a.pan b.pan c.pan
```

See also

logic

C++ prototype

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

Version française

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

Author: Régis Clouard

pareaclosing

Performs area closing (maxima killer).

Synopsis

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

Description

pareaclosing removes dark objects whose area is higher than the specified *area*.

The algorithm presented in a naive way consists in: thresholding image at each gray level and keeping regions whose area is higher than the specified *area*. The final result is the addition of the result at each level.

Parameters

- *connexity* specifies the relationship between a pixel and its neighbors. It is an integer from: 4 or 8 for 2D or 6 or 26 for 3D.
- *area* specifies the maximum area size to be removed.

Inputs

- *im_in*: a gray level 2D image.

Outputs

- *im_out*: an image of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Closes areas of the tangram pieces lower than 500 pixels:

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

See also

Morphology, pareaopening

C++ prototype

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

Version française

Fermeture aérolaire (tueur de surface sombre).

Author: Régis Clouard

pareadisorderfactor

Measures area disorder factor.

Synopsis

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

Description

pareadisorderfactor measures the area disorder factor of the region map *rg_in*.

This factor measures the regions size homogeneity. It is calculated as follows from all region areas size:

AD=1-1/(1+*standard-deviation(area)/mean(area)*).

This factor is a value between [0..1]. 0.0 characterizes homogeneous area sizes while 1.0 characterizes completely heterogeneous sizes.

This value can be get from the operator pstatus.

Inputs

- *rg_in*: a region map.

Result

Returns the value of the area disorder factor.

Examples

Displays area disorder factor of the region map yielded by a simple binarization of tangram.pan:

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

See also

Region Features Extraction

C++ prototype

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

Version française

Calcul du désordre surfacique d’une carte de régions.

Author: François Angot

pareaopening

Performs area opening (minima killer).

Synopsis

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

Description

pareaopening removes clear objects whose area is higher than the specified *area*.

The algorithm presented in a naive way consists in: thresholding image at each gray level and removing regions whose area is lower than the specified *area*. The final result is the addition of the result at each level.

Parameters

- *connexity* specifies the relationship between a pixel and its neighbors. It is an integer from: 4 or 8 for 2D or 6 or 26 for 3D.
- *area* specifies the maximum area size to be preserved.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Opens areas of the tangram pieces lower than 500 pixels:

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

See also

Morphology, pareaclosing

C++ prototype

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

Version française

Ouverture aérolaire (tueur de surface claire).

Author: Régis Clouard

pareaselection

Selects regions from area size.

Synopsis

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

Description

pareaselection selects regions from their area size. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

The area is calculated from the number of pixels included in the region and on the boundary. The algorithm uses one half pixel for concavity. For example, the area is 10 pixels for the following region (`8 + 4*0.5`):

```
 xx
xxxx
 xx
```

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is an integer defined in pixel unit.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *rg_out*: a 2D region map.

Result

Returns the number of selected regions.

Examples

Selects all regions with area size = 50 pixels:

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

See also

Region

C++ prototype

```
Errc PAreaSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, Ulong threshold );
```

Version française

Sélection de régions sur leur valeur de surface.

Author: Régis Clouard

parray2array

Converts array type.

Synopsis

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

Description

parray2array creates the collection *col_out* with all items of the input collection *col_in* except that the array *name* has been converted to the new *type*.

The conversion of all item values is done by using the C casting convention.

Parameters

- *name* is the name of the array to be converted. It is a string without blank character.
- *type* is the name of the type `Char, Uchar, Short, Ushort, Long, Ulong, Float, Double`.

Inputs

- *col_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Segments the tangram.pan image thanks to a k-means classification of the pixels based on mean and variance features:

```
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
```

See also

Array

C++ prototype

```
Errc Array2Array( Collection &col_in, const std::string &name, const
std::string &type );
```

Version française

Conversion du type d’une vecteur dans une collection.

Author: Alexandre Duret-Lutz

parray2im

Converts pixel array to image.

Synopsis

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

Description

parray2img2d builds the image *im_out* with size *w*x*h*x*d* from the value of the array *name* in the collection *im_out*.

If the array contains 1 vector then the output image is grayscale image. If the collection contains 3 vectors then the output is a color image.

Parameters

- *d*,*h* and *w* specify the size respectively the depth, the height and the width of the output image. For 2D image, d = 0.
- *name* is the name of the vector into the collection.

Inputs

- *col_in*: a collection.

Outputs

- *im_out*: an image.

Result

Returns SUCCESS or FAILURE.

Examples

Builds the 2D image a.pan from the vector foo in the collection col.pan

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

See also

Casting, pim2array

C++ prototype

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

Version française

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

Author: Alexandre Duret-Lutz

parrayargmax

Calculates the maximum values between arrays.

Synopsis

parrayargmax *name_in name_out* [*col_in*|-] [*col_out*|-]

Description

parrayargmax builds the array *name_out* in the output collection *col_out* with the indexes of the array that contains the maximum values between item at the same index. The input collection is supposed to contain arrays `name_in.1` ... `name_in.n` with the same size. The result is a new array *name_out* where each item `name_out[i]` is set with the number of the array that contains the maximum value between each array `name_in.1` ... `name_in.n` at the same index i.

Parameters

- *name_in* is the base name of the input arrays. It is a string without blank character.
- *name_out* is the name of output array.

Inputs

- *col_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

See also

Array

C++ prototype

```
Errc PArrayArgMax( const Collection &col_in, Collection &col_out,
const std::string &name_in, const std::string &name_out );
```

Version française

Extraction des valeurs maxima entre plusieurs tableaux.

Author: Alexandre Duret-Lutz

parraycovarmat

Calculates covariance matrix of arrays items.

Synopsis

parraycovarmat *name_in name_out* [*col_in*|-] [*col_out*|-]

Description

parraycovarmat calculates:

- the covariance matrix A,
- the inverse covariance matrix A^{-1} ,
- the determinant, and the mean array

from a set of arrays in the input collection *col_in*.

If there exists n arrays of p values each, the input collection *col_in* must contains p arrays name_in.1, name_in.2, ..., name_in.p of n float values each.

Parameters

- *name_in* specifies the base name of the array from which the covariance matrix will be calculated. It is a string without blank character.
- *name_out* is the base name of the output arrays:
  - name_out.mat : p*p covariance matrix.
  - name_out.inv : p*p inverse covariance matrix.
  - name_out.det : the determinant.
  - name_out.mean : mean array of p values.

Inputs

- *col_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Classifies beans into the jellybean.pan image from sample of each bean stored in the directory ’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
```

See also

Array

C++ prototype

```
Errc PArrayCovarMat( const Collection &col_in, , Collection
&col_out, const std::string &name_in, const std::string &name_out );
```

Version française

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

Author: Alexandre Duret-Lutz

parrayeuclideannorm

Calculates euclidean norm of arrays items.

Synopsis

parrayeuclideannorm *name_in name_out* [*col_in*|-] [*col_out*|-]

Description

parrayeuclideannorm calculates the euclidean norm of each array named `name_in.1`, `name_in.2`, ... , `name_in.n` in the input collection *col_in*. The result is a new array *name_out* in the output collection *col_out* in which each item contains the euclidean value of the related arrays (eg., `name_out[5]` = euclidean norm of `name_in.5`).

Parameters

- *name_in* is the base name of arrays in the input collection. It is a string without blank character.
- *name_out* is the name of the array in the output collection that contains the euclidean norm of each input arrays.

Inputs

- *im_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

See also

Array

C++ prototype

```
Errc PArrayEuclideanNorm( const Collection &col_in, Collection &col_out
                          const std::string &name_in, const std::string &name_out );
```

Version française

Calcul de la norme euclidienne de vecteurs.

Author: Alexandre Duret-Lutz

parraygetvalue

Gets value from array.

Synopsis

parraygetvalue *index name* [*col_in*|-]

Description

parraygetvalue returns the value of item at the specified *index* in the array *name* of the input collection *col_in*.

The result can then be get by the operator **pstatus**.

Parameters

- *name* is the name of the array in the collection. It is string without blank character.
- *index* is the rank of the item in the array. It is an integer from 0 to the number of item in the array -1.

Inputs

- *col_in*: a collection.

Result

Returns the value of the item or FAILURE.

Examples

See also

Array

C++ prototype

```
Errc PArrayGetValue( Collection &col_in, const std::String &name,
int index );
```

Version française

Extraction de la valeur d’un élément d’un vecteur dans une collection.

Author: Régis Clouard

parraymean

Calculates mean of array items.

Synopsis

parraymean *name_in name_out* [*col_in*|-] [*col_out*|-]

Description

parraymean calculates the mean of the items of the *name_in* array in the input collection *col_in*. The result is stored in the output collection *col_out* as the value named *name_out*.

Parameters

- *name_in* is the name of the array in the input collection.
- *name_out* is the name of the value in the output collection that stores the mean.

Inputs

- *col_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE

Examples

See also

Array

C++ prototype

```
Errc PArrayMean( const Collection &col_in, Collection &col_out
                 const std::string &name_in, const std::string &name_out );
```

Version française

Calcul des moyennes des valeurs de vecteurs.

Author: Alexandre Duret-Lutz

parraynorm

Performs normalization between 0 and 1 of items of one array.

Synopsis

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

Description

arraynorm creates the collection *col_out* with the normalized versions of the array *name* of the input collection *col_in*. The array in the output collection is converted to Double, and each item has been divided by the maximum value of the related type in the input array. For example, each item of Uchar array is divided by 255. The result values are between 0 and 1 for unsigned type or between -1 and 1 for signed type.

Parameters

- *name* is the name of the array to be normalized. It is a string without blank character.

Inputs

- *col_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

See also

Array

C++ prototype

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

Version française

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

Author: Alexandre Duret-Lutz

parraysize

Returns the size of array in a collection.

Synopsis

parraysize *name* [*col_in*|-]

Description

parraysize returns the size of the array *name* in the input collection *col_in*. The value can then be get by operator pstatus.

Parameters

- *name* is the name of the array in the collection. It is a string without blank character.

Inputs

- *col_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns an integer or FAILURE in case of bad input.

Examples

Classifies beans into the jellybean.pan image from sample of each bean stored in the directory ’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
```

See also

Array

C++ prototype

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

Version française

Retourne la taille d’un vecteur dans une collection.

Author: Alexandre Duret-Lutz

parraysnorm

Performs normalization between 0 and 1 of items of several arrays.

Synopsis

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

Description

parraysnorm creates the collection *col_out* with normalized versions of arrays of the input collection *col_in* `name.1`, `name.2`, ..., `name.n`. Arrays in the output collection are converted to Double, and each item has been divided by the maximum value of the related type in the input arrays. For example, each item of Uchar array is divided by 255. The result values are between 0 and 1 for unsigned type or between -1 and 1 for signed type.

Parameters

- *name* is the base name of the arrays to be normalized (name.1, name.2, ..). It is string without blank character.

Inputs

- *col_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Segments the tangram.pan image thanks to a k-means classification of the pixels based on mean and variance features:

```
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
```

See also

Array

C++ prototype

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

Version française

Normalisation des valeurs de plusieurs vecteurs entre 0 et 1.

Author: Alexandre Duret-Lutz

pbarbremoval

Removes barbs from length.

Synopsis

pbarbremoval *relation length* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pbarbremoval removes barbs from their length value. A barb is a chain of connected non null pixels with 1 pixel thickness that begins with an end point (a point with only one neighbor) and that ends at a junction (a point with more than two neighbors):

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

The parameter *relation* specifies the relation order to the *length* value that is used to select or not a barb. All other contour points are kept in the output image.

Warning: if the contour is not 1 pixel thickness, the operator may have unpredictable behavior. It might be necessary to use the operator ppostthinning to guaranty 1 pixel thickness.

Parameters

- *relation* is an integer from [-2,2] which specifies the relation order to the *length* value:
  - *relation* = 2: barbs ≥ *length*.
  - *relation* = 1: barbs > *length*.
  - *relation* = 0: barbs = *length*.
  - *relation* = -1: barbs < *length*.
  - *relation* = -2: barbs ≤ *length*.
- *length* is an integer defined in pixel unit.

Inputs

- *im_in*: a unsigned char graycale image (Img2duc or Img3duc).

Outputs

- *im_out*: an image of the same type as the input image.

Result

The number of removed barbs.

Examples

Removes barbs from contours yielded by a simple edge detection of tangram.pan:

```
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
```

See also

Contour

C++ prototype

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

Version française

Suppression des barbules sur leur longueur.

Author: Régis Clouard

pbellrescale

Performs a rescaling of image using the Bell algorithm.

Synopsis

pbellrescale *zoomx zoomy zoomyz* [*im_in*|-] [*im_out*|-]

Description

pbellrescale uses a convolution kernel to interpolate the pixel of the input image *im_in* in order to calculate the pixel value of the output image *im_out*. The interpolation consists in weigthing the input pixels influence on the output pixels. The weights are relative to the position of the output pixels and are given by the Bell algorithm:

```
| 0.75-sqr(x) if -0.5
```

pbetagraph

Builds the beta graph.

Synopsis

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

Description

pbetagraph builds the beta graph of the input graph *gr_in*. A beta graph is a graph where edges that are considered as too long are cut. The principle is to cut an edge if the circumcircle centered on the node contains an another node. The circumcircle between a node i and a node j is defined as follows:

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

where distance is the euclidean distance, and *beta* specifies the size of the intersection circle:

- if *beta*=1 the result is the Gabriel graph.
- if *beta*=2 the result is Relative Neighbor Graph.

The node values are kept in the output graph, whereas the edge weight values are set to 1.

Parameters

- *beta* is an integer from [0..2] that specifies the radius of the intersection circle.

Inputs

- *gr_in*: a graph.

Outputs

- *gr_out*: a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Calculates the Delaunay graph from the centers of mass of tangram pieces and then extracts the 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
```

See also

Graph

C++ prototype

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

Version française

Construction du béta-graphe d’un graphe.

Author: François Angot

pbicubicrescale

Performs an affine rescaling of 2D image using the bicubic interpolation.

Synopsis

pbicubicrescale *zoomx zoomy zoomyz* [*im_in*|-] [*im_out*|-]

Description

pbicubicrescale changes magnification of the input image by a factor *zoomx* along the x axis, *zoomy* along the y axis and *zoomz* along the z axis (for 3D images). The image is enlarged along an axis if the zoom factor is > 1 and is shrunk if the zoom factor is >0 and <1.

This version uses the bicubic interpolation. For bicubic interpolation, the output pixel value is a weighted average\ of pixels in the nearest 4-by-4 neighborhood.

This operator needs a long execution time. Thus it can only be used for 2D images. To rescale region map or graph or 3D image, use the operator prescale.

Parameters

- *zoomx*, *zoomy*, *zoomz* are positive real values.

  if a zoom factor is > 1 then the image is enlarged along the related axis.

  if a zoom factor is < 1 then the image is shrunk along the related axis.
  zoomz is ignored since this operator only works on 2D images.

Inputs

- *im_in*: an 2D image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

- Enlarges the tangram.pan image by a factor 2:

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

- Shrinks the tangram.pan image by a factor 2:

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

See also

Transformation, plinearrescale, prescale

C++ prototype

```
Errc PBicubicRescale( const Img2duc &im_in, Img2duc &im_out, float zoomy, float zoomx );
```

Version française

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

Author: Régis Clouard

pbinarization

Performs binary thresholding on image and graph.

Synopsis

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

Description

pbinarization classifies pixels of the input image *im_in* into 2 clusters. Pixels values that are lower than *low* or greater than *high* are set to 0; other values are set to 255:

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

If *high* is lower than *low* then **pbinarization** performs an inverse thresholding:

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

For multispectral and color images, the threshold is compared with the min or max pixel value of each band. For graph, **pbinarization** operates on graph nodes.

Parameters

- *low* and *high* specify the gray level bounds. Values are from the gray levels of the input image (eg. [0..255] for byte image, [-2147483648..+2147483648] for long integer image).

  Tip: If *high* is lower than *low* then **pbinarization** performs an inverse thresholding.

  Tip: If *high* is greater than the maximum gray level then *high* is set with the maximum value (respectively for *low*).

Inputs

- *im_in*: an image or a graph.

Outputs

- *im_out*: an image of bytes (Img2duc, Img3duc) or a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Segments the tangram.pan image in 2 classes, background and tangram pieces:

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

See also

Thresholding

C++ prototype

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

Version française

Seuillage binaire d’une image.

Author: Régis Clouard

pblend

Performs alpha blending of image or graph.

Synopsis

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

Description

pblend computes the alpha blending of the two inputs *im_in1* and *im_in2*.

If *im_in1* and *im_in2* are images then the new image *im_out* is built with the blending of each pixels:

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

The two inputs must be of the same type.

The output image *im_out* is of the same type as inputs.

For color or multispectral image, the blending is computed separately on each band.

If *im_in1* and *im_in2* are graphs then the new graph *im_out* is built with the blending of each node values.

Paramaters

- alpha is a real value between [0..1] which represents the ratio of *im_in1*.

Inputs

- *im_in1*: an image or a graph.
- *im_in2*: an object of the same type as *im_in1*.

Outputs

- *im_out*: an object of the same type as inputs.

Result

Returns SUCCESS or FAILURE.

Examples

Performs a mean operation between a.pan and b.pan:

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

See also

Arithmetic

C++ prototype

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

Version française

Mélange d’images ou de graphes.

Author: Régis Clouard

pblockmatching

Motion estimation between 2 images using block matching.

Synopsis

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

Description

pblockmatching builds a new image that contains the motion estimation between 2 images.The output image *im_out_dep* is a multispectral image containing a motion vector in each point: the first band contains the abscissa, and the second one the ordinate of the motion vector.

The motion estimation method using the "block matching" is to find a matching between each squared block of size *block_size* in the reference image *im_in_ref* with the same block in the destination image *im_in_dest*. The corresponding distination block is searched in a the neigbourhood of size *search_size*. The matching criteria is the destination block that mimizes the Sum of Square Differences (SSD):

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

Parameters

- *block_size*: block size. A common value is 16.
- *search_size*: radius of search for the corresponding block.
- *ssd_min*: below this threshold motion is not considered to be significant.

Inputs

- *im_in_ref*: a 2D reference image.
- *im_in_dest*: a 2D image.

Outputs

- *im_out_dep*: a 2D multispectral image (band 0: abscissa, band 1: ordinate)

Result

Return SUCCESS or FAILURE.

Examples

Estimates the motion between an image and its translated:

```
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
```

See Also

Motion

C++ prototype

```
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);
```

Version française

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

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

pbmp2pan

Converts BMP image file to Pandore image file.

Synopsis

pbmp2pan *im_in* [*im_out*|-]

Description

pbmp2pan converts a bitmap image file (BMP) to a Pandore image file. BMP files encode two types of images: gray level 2D image of bytes (Img2duc) and a color 2D image of bytes (Imc2duc).

Note: Only uncompressed BMP image can be converted. It might be necessary to use an another image converter to convert compressed BMP file to uncompressed BMP file.

Inputs

- *im_in*: a BMP image file.

Outputs

- *im_out*: a Pandore image file.

Result

Returns SUCCESS or FAILURE.

Examples

Converts BMP image to Pandore image

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

See also

Conversion, ppan2bmp

C++ prototype

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

Version française

Conversion d’une image BMP en image Pandore.

Author: Régis Clouard

pborsotti

Computes the goodness measure based on the number, area and variance of regions.

Synopsis

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

Description

pborsotti computes a goodness measure for quantitative evaluation of gray levels, color and multispectral image segmentation results as defined by M. Borsotti [*].

The measure is defined from three criteria:

- regions must be uniform and homogeneous;
- the region's interiors must be simple, without too many small holes;
- adjacent regions must present significantly different values for uniform characteristics.

The measure is computed as follows:

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

where

- A is the total area of regions.
- Ai is the area of the region i.
- N is the number of regions.
- R(Ai) is the number of regions that have the same area than Ai.
- ei is defined as the sum of euclidean distances between the color vectors of the pixel of the region i and the color vector attributed to region i.

The previous equation is composed of three terms:

1. a normalization factor that takes into account the size of the image;
2. a penalization factor for under-segmented regions;
3. the sum is composed of two parts:
   - penalization for small regions or heterogeneous of the regions;
   - penalization for region with the same size (over-segmentation).

The smaller the value of the Borsotti's measure is, the better the segmentation result should be.

Caution: Regions with label=0 are not considered for computing.

Inputs

- *rg_in*: a region map.
- *im_in*: an image.

Result

Returns a positive real value.
(Use `pstatus` to get this value).

Examples

Computes the borsotti measure for a simple binarization segmentation process:

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

See also

Evaluation

C++ prototype

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

Version française

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

Reference

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

Author: Régis Clouard

pboundary

Locates region boundary.

Synopsis

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

Description

pboundary builds a binary image with pixels that separate two regions. A pixel is a boundary point if at least one of its neighbors has a different value than itself.

A boundary is a closed contour with a width of 2 pixels because it cannot be located between two pixels. One pixel is inside the region and one pixel is outside the region. Each boundary point is represented with a value of 255 in the output image. The output image is an unsigned char grayscale image (Img2duc or Img3duc).

For region map, only regions with label greater than 0 are considered.

For graph, nodes with 0 or 1 neighbor are considered as boundary points.

Parameters

- *connexity* defines the neighborhood relation: 4, 8 for 2D, or 6 or 26 for 3D.
  This parameter is ignored for graphs.

Inputs

- *im_in*: a grayscale image, a region map or a graph.

Outputs

- *im_out*: a binary image (Img2duc or Img3duc).

Result

Returns SUCCESS or FAILURE.

Examples

Locates the boundary in the region map a.pan (add a boundary on the border):

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

See also

Region

C++ prototype

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

Version française

Localisation des points de frontière des régions.

Author: Régis Clouard

pboundarylabeling

Performs region labeling from boundary.

Synopsis

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

Description

pboundarylabeling builds regions from a set of connected pixels surrounded by a closed contour. A closed contour is a chain of 8-connected non null pixels that describes a loop.

A region is built as a set of connected pixels with the same label. The label value is set randomly from 1 to the total number of regions.

Notice: Pixels with label value = 0 are not considered as part of region. Thus, the region 0 does not exist.

Inputs

- *im_in*: a 2D grayscale image that contains contours.

Outputs

- *rg_out*: a region map.

Result

Returns the number of regions.

Examples

Closes edges and labels related regions:

```
psobel tangram.pan b.pan
pbinarization 34 1e30 b.pan c.pan
pskeletonization c.pan d.pan
pboundarylabeling d.pan out.pan
```

See also

Segmentation, plabeling

C++ prototype

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

Version française

Etiquetage en régions d’une image de contours fermés.

Author: Régis Clouard

pboundarymerging

Performs priority region merging based on uniformity criterion.

Synopsis

pboundarymerging *number threshold* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

pboundarymerging merges connected regions of the input image *rg_in* if the difference between the boundary contrast criterion of the region is lower than the specified *threshold*.

Two regions are connected if there exists a link between the related nodes in the input graph *gr_in*.

The principle of the algorithm is as follows:

- For each region of the input region map *rg_in*:
- If the difference between the criterion value of the two connected regions ≤ *threshold* then merge them into one region.

The algorithm uses the priority merging that consists in merging regions with the lower difference.

The output region map *reg_out* defines the new regions and the output graph *gr_out* defines the new relationship between regions.

The boundary contrast is calculated as follows:

```
contrast(R1,R2)= 1/N * SUM(MAX(C(s,t), t in V(s) and t in R2 and s in R1))
```

where C(s,t)= | im_in[s] - im_in[t] |
and N is the number of boundary pixels
and V(s) is the neighbors of pixel s.

Parameters

- *number* specifies the number of allowed merging. If *number* = -1 then all possible merging are done.

- *threshold* specifies the maximum difference allowed between two regions to decide to merge them. Values are from the gray scale of the input image.

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.
- *gr_out*: a graph.

Result

Returns the number of merging.

Examples

Merges regions yielded by a quadtree splitting process:

```
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
```

See also

Segmentation

C++ prototype

```
Errc PBoundaryMerging( const Reg2d &rg_in, const Graph2d &gr_in,
const Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, long number,
float threshold );
```

Version française

Fusion prioritaire de régions selon le contraste aux frontières.

Author: Laurent Quesnel

pboundingbox

Builds the bounding box around regions.

Synopsis

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

Description

pboundingbox builds a new region map *im_out* with the bounding box around each region of the input region map *im_in*. The output region map is composed of rectangles.

The labels of the bounding box is the same as the label of the related region.

Warning: It might occur some overlapping between bounding boxes. Therefore, a bounding box can hide another.

Inputs

- *rg_in*: a region map.

Outputs

- *rg_out*: a region map.

Result

Returns SUCCESS or FAILURE.

Examples

Builds the bounding boxes around the region of rin.pan

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

See also

Region

C++ prototype

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

Version française

Calcul du rectangle exinscrit des régions.

Author: François Angot

pbutterworthfilter

Designs lowpass, highpass, bandpass or bandreject Butterworth filter.

Synopsis

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

Description

pbutterworthfilter designs either a lowpass, highpass, bandpass or bandreject Butterworth filter. If *ndep<1* the filter *im_out* is a 2D float image with size nrow*ncol otherwise the filter *im_out* is a 3D float image with size ndep*nrow*ncol.

The Butterworth lowpass filter cuts off high-frequency components of the Fourier transform that are at a distance greater than a specified distance D0 (the *cutoff* value) from the origin of the centered transform.

The type of filter is given by both parameters *highpass* and *cutin*:

- highpass=0 and cutin=0 : lowpass filter
- highpass=1 and cutin=0 : highpass filter
- highpass=0 and cutin=1 : bandreject filter
- highpass=1 and cutin=1 : bandpass filter

The transfer function for a 2D Butterworth lowpass filter of the given *order* n and with *cutoff* frequency at distance D0 from the origin is defined as:

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

where D(u,v) is the distance of point (u,v) from the origin:

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

where N is the number of rows and M is the number of columns.

The transfert function for a Butterworth highpass is defined as:

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

The transfer function for a band reject is defined as:

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

where W is the bands width = cutoff-cutin and D0 is the radius=(cutin+cutoff)/2.

Parameters

- *ncol, nrow, ndep* specify the size of the output image. If *ndep*<1 then the output image *im_out* is a 2D image otherwise a 3D image.
- *highpass* is used in conjunction with the *cutin* parameter. It specifies the type of the filter:
  - highpass=0 and cutin=0 : lowpass filter
  - highpass=1 and cutin=0 : highpass filter
  - highpass=0 and cutin=1 : bandreject filter
  - highpass=1 and cutin=1 : bandpass filter
- *cutin* is the cut in frequency of the filter D0 in case of bandreject or bandpass filter. In this case, the band width=cutoff-cutin and D0=(cutoff+cutin)/2.
- *cutoff* is the cut off frequency of the filter D0. It must be a positive real value in the interval]0.. sqrt(M*m+N*n)/2]. It corresponds to an euclidean distance from the center of the image. The higher cutoff is, the lower the lowpass is or the higher the highpass is.
- *order* is the order of the filter. The higher order is the sharper the transition is. It must be an integer >= 1. A typical value is 1.

Outputs

- *im_out*: a float image (Img2dsf or Img3dsf).

Result

Returns SUCCESS or FAILURE in case of bad parameter values.

Examples

Performs Butterworth lowpass filtering:

```
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
```

Performs Butterworth highpass filtering:

```
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
```

Performs Butterworth bandreject filtering:

```
  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
```

Performs Butterworth bandpass filtering:

```
  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
```

See also

Frequency, pifft pfftshift

C++ prototype

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

Version française

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

Author: Régis Clouard

pcenterofmass

Locates the center of mass of regions.

Synopsis

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

Description

pcenterofmass builds a new region map *im_out* with the center of mass of each region of the input region map *im_in*. A center of mass is represented as a point in the output region map with the same label value than the related region.

The center of mass of a region of size N is calculated as follows:

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

Warning: It might occur some overlapping between centers of mass because there are not necessarily located into the region. Therefore, a center of mass can hide another.

Inputs

- *rg_in*: a region map.

Outputs

- *rg_out*: a region map.

Result

Returns SUCCESS or FAILURE.

Examples

Locates the center of mass of regions in the region map rin.pan:

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

See also

Region

C++ prototype

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

Version française

Localisation des centres de gravité de régions.

Author: Régis Clouard

pchanda

Performs multi-thresholding on image using Chanda algorithm.

Synopsis

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

Description

pchanda classifies the input image pixels into a small number of clusters according to their value. Every pixel p of the input image is assigned to a cluster identified by the related threshold value:

```
if threshold[k-1]<im_out[p]<=threshold[k].
then im_out[p]=threshold[k]
```

The last threshold is equal to 255.

The number of clusters and the value of the thresholds are determined from the Chanda et al. algorithm. It is based on the measure of the average contrast value. For each gray level i the average contrast value is:

$$\text{contrast}(i) = \frac{\text{SUM}(\text{SUM}((Tkl)*(Tkl))}{\text{SUM}(\text{SUM}(Tkl))} + \frac{\text{SUM}(\text{SUM}((Tpq * Tpq)))}{\text{SUM}(\text{SUM}(Tpq))}$$

with k in [0..i], l in [i+1..N-1], p in [i+1..N-1] and q in [0..i].

The co-occurrence matrix Tkl contains the number of times the gray level k is the neighbor of the gray level l considering the neighborhood Nxy={ (x,y+1) }.

Then the thresholds are located as regional maxima of the average contrast function. The maxima are searched in the space of *length* gray levels around the gray level i.

Notice: This operator can only work on grayscale image of bytes.

Parameters

- *length* defined the length of the search space of the regional maxima. It is defined in gray level unit. The greater is the length, the less there are thresholds. A typical value is 10.

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).

Outputs

- *im_out*: a grayscale image of bytes (Img2duc, Img3duc).

Result

Returns the number of thresholds.

Examples

Segments tangram.pan and displays the number of thresholds:

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

See also

Thresholding

C++ prototype

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

Version française

Multiseuillage de l’image par analyse de la matrice de co-occurences selon Chanda.

Reference

B. Chanda, Chauduri and Majumder, "On Image Enhancement and threshold selection using the gray lavel co-occurence matrix", *Pattern Recognition Letter*, Vol.3, No. 4, pp. 243-251, 1985.

Author: Régis Clouard

pcliparea

Select rectangular region area of image, region map or graph.

Synopsis

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

Description

pcliparea select a rectangular region area of the input *im_in*. Values outside the rectangle are set to 0, other values are copied to the output *im_out*.

Parameters

- *x*, *y*, *z* specify the cordinates and *width*, *height*, *depth* the size of the rectangle. For 2D objects, *z* and *depth* must be given but are ignored.

Inputs

- *im_in*: an image, a region map, or a graph.

Outputs

- *im_out*: an object of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Select area (10,10,50,50) from the tangram.pan image :

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

See also

Utility

C++ prototype

```
Errc PClipArea( const Imxg2duc &ims, Img2duc &imd, const int z,
const int y, const int x, int depth, int height, int width) {
```

Version française

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

Author: Régis Clouard

pclipvalues

Clips pixel values inside the specified range.

Synopsis

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

Description

pclipvalues restricts the pixel values of the input image *im_in* to the range defined by the parameters [*low*..*high*]. Pixels with a value lower than *low* will be set to *low*; pixels with a value higher than *high* will be set to *high*.

More formally, *im_out* is built using the following algorithm for each 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];
```

Parameters

- *low* and *high* specifies the range of the output pixel values. They are related to the input type image (for example Img2duc [0..255],Img2dsl [-2147483648..+2147483648]).

  Note: if *min* > *max* then min and max are set respectively with the minimum and maximum values of the input image type. (For example, 0 and 255 for Uchar images or +2147483648 for Img2dsl).

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image with the same properties as im_in.

Result

SUCCESS or FAILURE in case of invalid parameter values.

Examples

Contrast sharpening of the tangram.pan image using the unsharp masking technique. The sharpened image is built by adding a highpass filtering image of the initial image tangram.pan. The highpass image is built by subtracting from the tangram.pan image a blurred version of itself. At the end, only pixel values within the range [0..255] are kept in the final result. All the processing is done with float images.

```
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
```

See also

Arithmetic

C++ prototype

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

Version française

Ecrêtage des valeurs de pixels

Author: Régis Clouard

pclosedcontourselection

Selects closed contour from length.

Synopsis

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

Description

pclosedcontourselection selects closed contours from their length. A closed contour is a chain of connected non null pixels with 1 pixel thickness that forms a loop.

The parameter *relation* specifies the relation order to the *length* value that is used to select or not a contour.

Warning: if the contour is not 1 pixel thickness, the operator may have unpredictable behavior. It might be necessary to use the operator ppostthinning to guaranty 1 pixel thickness.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *length* value:
  - *relation* = 3: closed contours with the maximum length.
  - *relation* = 2: closed contours >= *length*.
  - *relation* = 1: closed contours > *length*.
  - *relation* = 0: closed contours = *length*.
  - *relation* = -1: closed contours < *length*.
  - *relation* = -2: closed contours <= *length*.
  - *relation* = -3: closed contours with the minimum length.
- *length* is an integer defined in pixel unit.

Inputs

- *im_in*: a 2D or 3D grayscale image of unsigned char (Img2duc or Img2duc).

Outputs

- *im_in*: an unsigned char image (Img2duc or Img3duc).

Result

The number of selected contours.

Examples

Selects the longest closed contours from contours yielded by a simple edge detection of tangram.pan:

```
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
```

See also

Contour

C++ prototype

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

Version française

Sélection des chaînes de contours fermées sur leur longueur.

Author: Régis Clouard

pcol2txt

Converts collection to text file.

Synopsis

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

Description

pcol2txt converts the contents of the input collection *col_in* into a text file. Only numerical values are printed, Pandore object are just listed.

The name of the text file is optional. If it is omitted, the result is printed on the standard output. This operator is a mean to display the contents of a collection.

Notice: Uchar and Char arrays are printed as tiny integer arrays. Therefore, strings (Array:Char) are printed as a sequence of ascii code values (where A = 65, B=66 ...).

Inputs

- *col_in*: a collection.

Outputs

- *file_out*: a text file.

Result

Returns SUCCESS or FAILURE.

Examples

Displays the contents of the collection col.pan:

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

See also

Collection, ptxt2col

C++ prototype

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

Version française

Conversion d’une collection en un fichier texte.

Author: Alexandre Duret-Lutz

pcolcatenateitem

Catenates two collections.

Synopsis

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

Description

pcolcatenateitem creates the collection *col_out* from items of the two input collections *col_in1* and *col_in2*.

If an item has the same name in the two collections and the type is compatible then an array is created with the concatenation of the values of the two collections otherwise the item values of the first collection *col_in1* are preferred.

Inputs

- *col_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Catenates the 2 collections c1.pan and c2.pan in the collection col.pan and then extracts the two images in file foo.pan and bar.pan:

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

See also

Collection

C++ prototype

```
Errc PColCatenateItem( const Collection &col_in_out, const
Collection &c2 );
```

Version française

Concaténation des éléments de deux collections.

Author: Alexandre Duret-Lutz

pcolgetimages

Extracts images from collection.

Synopsis

pcolgetimages [*col_in*|-]

Description

pcolgetimages extracts all images from the specified collection *col_in*. Each image is saved in a file named from the related name into the collection suffixed by ".pan".

For example, if the command: "pfile col.pan" returns :

```
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
```

then the command "pcolgetimages col.pan" builds :

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

Operator pcolgetobject can be used to get only one image from the input collection.

Inputs

- *col_in*: a collection.

Result

Returns SUCCESS or FAILURE if at least one image cannot be saved.

Examples

Catenates the 2 collections c1.pan and c2.pan in the collection col.pan and then extracts the two images in file foo.pan and bar.pan:

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

See also

Collection, pcolgetobject

C++ prototype

```
Errc ColGetImages( const Collection &col_in ) ;
```

Version française

Extraction des images d’une collection.

Author: Nicolas Briand

pcolgetobject

Gets Pandore object from collection.

Synopsis

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

Description

pcolgetobject extracts the Pandore object named *name* from the input collection *col_in* to create the output file *obj_out*. Pandore object can be image, region map, graph or even collection.

Parameters

- *name* is the name of the object in the collection. Is it a string without blank character.

Inputs

- *col_in*: a collection.

Outputs

- *obj_out*: a Pandore file.

Result

Returns SUCCESS or FAILURE if the object does not exist or it is not a Pandore object.

Examples

Extracts the image named foo in the collection col.pan into the file a.pan:

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

See also

Collection, pcolgetimages

C++ prototype

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

Version française

Extraction d’un objet Pandore d’une collection.

Author: Alexandre Duret-Lutz

pcolgetvalue

Gets numerical value from a collection.

Synopsis

pcolgetvalue *name* [*col_in*|-]

Description

pcolgetvalue returns the value of the item *name* from the collection *col_in*. The item must be a basic type: Char, Uchar, Short, Ushort, Long, SLong, Float or Double.

The value can then be get with the operator **pstatus**.

Parameters

- *name* is a string without blank character.

Inputs

- *col_in*: a collection.

Result

Returns the numeric value or FAILURE if the specified item doesn't exist.

Examples

Adds the float value 10.5 in the collection col.pan with the name "foo" and chekcs if the value is in the collection:

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

See also

Collection, pcolsetvalue

C++ prototype

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

Version française

Extraction de la valeur d’un élément numérique dans une collection.

Author: Régis Clouard

pcolorcube

Creates the RGB cube of image color repartition.

Synopsis

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

Description

pcolorcube builds a synthetic image that visualizes the RGB cube of the color repartition of the input image *im_in*.

The cube is viewed in 2D according to the position of the observer specified by the coordinates (*x*,*y*,*z*) from the origin of the cube.

The main axis appears in the image as a white line.

Parameters

- *x*,*y* and *z* specify the position of the observer from the cube origin.

Inputs

- *im_in*: a color image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Displays the color cube of the parrot.pan image:

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

See also

Visualization

C++ prototype

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

Version française

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

Author: Olivier Lezoray

pcolorize

Colorization of regions with internal average value.

Synopsis

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

Description

pcolorize builds the output image *im_out* from the mean color of the regions of the input image *im_in* an specified in the region map *rg_in*. Each pixel of the output image is set with the mean color of the related region.

The output image is of the same type as the input image.

Inputs

- *im_in*: an image.
- *rg_in*: a region map.

Outputs

- *im_out*: an image of the same type as the input image.

Examples

Colorizes regions of the image reg.pan with their inner mean value:

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

Result

Returns SUCCESS or FAILURE.

See also

Visualization

C++ prototype

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

Version française

Colorization de régions à partir de sa valeur moyenne.

Author: Olivier Lezoray

pcolremoveitem

Removes item from collection.

Synopsis

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

Description

pcolremoveitem deletes the specified item *name* from the input collection *col_in*. The result collection is saved in the output collection *col_out*.

Parameters

- *name* is the name of the input item. It is a string without blank character.

Inputs

- *col_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE if the specified item does not exist.

Examples

Removes the item "foo" from the collection col.pan:

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

See also

Collection

C++ prototype

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

Version française

Suppression d’un élément dans une collection.

Author: Alexandre Duret-Lutz

pcolprenameitem

Renames item in collection.

Synopsis

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

Description

pcolrenameitem renames the item named *old_name* in the input collection *col_in* to *new_name* in the output collection *col_out*.

Parameters

- *old_name* is the name of the input item. It is a string without blank character.
- *new_name* is the name of the output item. It is a string without blank character.

Inputs

- *col_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Renames item "foo" by "bar":

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

See also

Collection

C++ prototype

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

Version française

Renommage d’un élément dans une collection.

Author: Alexandre Duret-Lutz

pcolsetobject

Sets Pandore object to collection.

Synopsis

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

Description

pcolsetobject adds a Pandore object to the input collection *col_in* with the specified *name*.

If the item named *name* already exists, the value is replaced by the new object.

Parameters

- *name* is the name of Pandore object in the collection. It is a string without blank character.

Inputs

- *col_in*: a collection.
- *obj_in*: a Pandore file.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Adds tangram.pan and parrot.pan to collection col.pan:

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

See also

Collection, pobject2col

C++ prototype

```
Errc PColSetObject( const Collection &col_in_out , PObject &obj_in,
const std::string &name );
```

Version française

Création d’une collection contenant un objet Pandore.

Author: Alexandre Duret-Lutz

pcolsetvalue

Sets numerical value to collection.

Synopsis

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

Description

pcolsetvalue adds the specified value *value* of type *type* to the input collection *col_in* with the specified *name*. The result collection is saved into *col_out*.

If the item named *name* already exist, the value is replaced by the new value *value*.

Parameters

- *name* is the name of the numerical value in the input collection. It is a string without blank character.
- *type* is a string that defines the numerical type among [Char, Uchar, Short, Ushort, Long, Ulong, Float, Double].
- *value* is a numerical value which type depends on the input image type.

Inputs

- *col_in*: a collection.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Adds the float value 10.5 in the collection col.pan with the name "foo" and verifies if the value is in the collection:

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

See also

Collection, pcolgetvalue

C++ prototype

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

Version française

Ajout de la valeur d’un élément numérique dans une collection.

Author: Régis Clouard

pcompactnessselection

Selects regions from compactness factor.

Synopsis

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

Description

pcompactnessselection selects regions from their compactness degree. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

Compactness represents the degree to which the shape region is compact. It is defined as the ratio of the area of a region to the area of a circle with the same perimeter. It is calculated as follows:

```
compactness = (4*PI*area) / (perimeter*perimeter)
```

For a circle, the compactness is 1.0, for a square, it is PI/4 and for an infinitely long and narrow shape, it is zero.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is a real value from [0..1*]. (*1 or more because of the roundness errors.)

Inputs

- *rg_in*: a region map.

Outputs

- *rg_out*: a region map.

Result

Returns the number of selected regions.

Examples

Selects regions with the highest compactness factor:

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

See also

Region

C++ prototype

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

Version française

Sélection de régions sur leur valeur de compacité.

Author: Régis Clouard

pcontentsdisplay

Displays the contents of Pandore object.

Synopsis

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

Description

pcontentsdisplay displays the contents of the input object *obj_in* on the terminal. Only non null values are displayed. Thanks to the masking operation (-m *mask*), it is possible to select object area to be displayed.

If *obj_in* is an image, it only displays non null pixel values.

If *obj_in* is a graph, it only displays each node values.

If *obj_in* is a collection, it only lists items in the collection and displays their memory size.

Inputs

- *obj_in*: a Pandore object.

Result

Returns SUCCESS or FAILURE.

Examples

Displays the contents of the entire tangram.pan image:

```
pcontentsdisplay tangram.pan
```

Displays the contents of the center part of the tangram.pan image (size 50x50) :

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

See also

Visualization

C++ prototype

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

Version française

Affichage du contenu d’un objet Pandore.

Author: François Angot

pcontourextensionconic

Performs cone-shaped extension of end points.

Synopsis

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

Description

pcontourextensionconic extends all contour end points by a cone shape defined by the size *length*. A contour chain is a sequence of connected non null pixels with 1 pixel thickness. An end point is a contour point with only 1 neighbor.

The extension of the end point is done with a cone of the specified *length* in the sense of the contour. For example, the 2D contour on the left is extended to the contour on the right if *length*=3.

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

Warning: if the contour is not 1 pixel thickness, the operator may have unpredictable behavior. It might be necessary to use the operator ppostthinning to guaranty 1 pixel thickness.

Parameters

- *length* is an integer that specifies the length of the cone in the sense of the contour.

Inputs

- *im_in*: a 2D or 3D grayscale image (Img2duc or Img3duc).

Outputs

- *im_out*: an image of the same type as the input image.

Result

The number of end points.

Examples

Closes contours yielded by a simple edge detection of tangram.pan by using an extension of the end points:

```
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
```

See also

Contour

C++ prototype

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

Version française

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

Author: Arnaud Renouf

pcontourextensionrect

Performs rectangular-shaped extension of end points.

Synopsis

pcontourextensionrect *length height* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pcontourextensionrect extends all contour end points by a rectangle (or a parallelogram in 3D) defined by the size *length* x *height* (or *length* x *height* x *height* for 3D). A contour chain is a sequence of connected non null pixels with 1 pixel thickness. A end point is a contour point with only 1 neighbor. For a 3D extension, the depth is equal to the *height*.

For example, the 2D contour on the left is extended to the contour on the right if *length*=4 and *height*=1.

```
                  xxxx
xxxx     ->   xxxxxxxx
                  xxxx
```

For the current 3D version, height is not taken into account. The extension is only 1 pixel height and depth.

Warning: if the contour is not 1 pixel thickness, the operator may have unpredictable behavior. It might be necessary to use the operator ppostthinning to guaranty 1 pixel thickness.

Parameters

- *length* is the length, counted in pixel unit, of the extension in the sense of the contour.
- *height* is the thickness, counted in pixel unit, of the extension in the orthogonal sense of the contour.

Inputs

- *im_in*: a 2D or 3D grayscale image (Img2duc or Img3duc).

Outputs

- *im_out*: an image of the same type as the input image.

Result

The number of end points.

Examples

Closes contours yielded by a simple edge detection of tangram.pan by using an extension of the end points:

```
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
```

See also

Contour

C++ prototype

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

Version française

Extension des points terminaux dans la direction du contour.

Author: Régis Clouard

pcontourselection

Selects contour chain from length.

Synopsis

pcontourselection *relation length* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pcontourselection selects open contours from their length. A contour chain is a sequence of connected non null pixels with 1 pixel thickness that begins and ends at an end point or a junction. A close contour, a barb and a line are considered as piecewise contour chains:

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

The parameter *relation* specifies the relation order to the *length* value that is used to select or not a contour.

Warning: if the contour is not 1 pixel thickness, the operator may have unpredictable behavior. It might be necessary to use the operator ppostthinning to guaranty 1 pixel thickness.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *length* value:
  - *relation* = 3: contours with the maximum length.
  - *relation* = 2: contours ≥ *length*.
  - *relation* = 1: contours > *length*.
  - *relation* = 0: contours = *length*.
  - *relation* = -1: contours < *length*.
  - *relation* = -2: contours ≤ *length*.
  - *relation* = -3: contours with the minimum length.
- *length* is an integer defined in pixel unit.

Inputs

- *im_in*: a unsigned char graycale image (Img2duc or Img3duc).

Outputs

- *im_out*: an image of the same type as the input image.

Result

The number of selected contours.

Examples

Selects contours with length greater than 5 pixels from contours yielded by a simple edge detection of tangram.pan:

```
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
```

See also

Contour

C++ prototype

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

Version française

Selection des chaînes de contours ouvertes sur leur longueur.

Author: Régis Clouard

pcontrast1quadtree

Performs quadtree (or octree) segmentation based on contrast criterion.

Synopsis

pcontrast1quadtree *threshold* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

pcontrast1quadtree segments the input image *im_in* into homogenous regions. Homogeneous regions are regions that have an inner contrast ≤ *threshold*.

The principle of the algorithm is as follows:

- At the begin consider the image as the first block.
- If the block violates the uniformity predicate (i.e. inner contrast ≤ *threshold*) then split the block into four equally sized sub-blocks and then apply the algorithm recursively on each sub-blocks.

Therefore, the result is composed of rectangular regions.

The contrast is calculated from:

```
    contrast(R)= 1/N * SUM(max(C(s,t), t in V(s) and t in R))
and C(s,t)= | im_in[s] - im_in[t] |
```

where N is the number of pixels of the region R.

For 3D image, the output region map is composed of octree regions.

Parameters

- *threshold* is the maximum contrast value to decide if a region is homogeneous or not. Values are from the gray scale of the input image *im_in* (eg., 0-255 for Uchar image).

Inputs

- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of regions.

Examples

Builds the quadtree of tangram.pan:

```
pcontrast1quadtree 10 tangram.pan a.pan
```

See also

Segmentation

C++ prototype

```
Errc PContrast1Quadtree( const Img2duc &im_in, Reg2d &rg_out, Uchar threshold );
```

Version française

Segmentation d’une image par quadtree (ou octree) selon le contraste.

Author: Laurent Quesnel

pcontrast1value

Measures the global contrast value of grayscale image or graph.

Synopsis

pcontrast1value [*im_in*|-] [*col_out*|-]

Description

pcontrastvalue measures the global contrast of the input image *im_in* or the input graph *im_in*.

The contrast is measured as follows:

```
C= SUM( MAX (C(s,t), C in V(s))) / N
and C(s,t) = |im_in[s] - im_in[t]| / (K-1)
```

where K is the number of gray levels, and N is the number of pixels.

For image, the contrast is measured from the pixel values.

For graph, the contrast is measured from the node values.

The contrast values for each band are stored in the collection *col_out*.

Inputs

- *im_in*: an image or a graph.

Outputs

- *col_out*: a collection of float values.

Result

Returns the global contrast value (for the first band only). This value can be get using operator **pstatus**.

Examples

Measures the global contrast of the tangram.pan (Unix version):

```
pcontrast1value tangram.pan col.pan
var=‘pstatus‘
echo "Contrast = $val"
```

Measures the global contrast of the tangram.pan (MsDos version):

```
pcontrast1value tangram.pan col.pan
call pstatus
call pset var
echo Contrast = %val%
```

See also

Image Features Extraction

C++ prototype

```
Float PContrast1Value( const Img2duc &im_in, Collection & col_out );
```

Version française

Calcul du contraste global d’une image ou d’un graphe.

Author: Régis Clouard

pcontrastaggregation

Performs pixel aggregation based on contrast criterion.

Synopsis

pcontrastaggregation *connexity threshold* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

pcontrastaggregation builds a new region map from aggregation of pixels to regions of the input region map *rg_in*. A pixel p is aggregated to a region R if:

- p is connected to the region R according to the specified *connexity*;
- |contrast(R)-contrast(R+im_in[p])| ≤ threshold.

The contrast is approximated by:

```
contrast(R)=MAX(r)-MIN(R)
```

The contrast of the region is not updated with the new pixel to avoid moving away too much from the initial situation. One prefer iterative executions of the operator to update the inner contrast. For example, operator can be iterated until pstatus returns 0.

The output region map *rg_out* has the same number of labels than the input region map.

Parameters

- *connexity* specifies the neighbor relation between pixel and region (4 or 8 for 2D; 6 or 26 for 3D).
- *threshold* specifies the maximum variance value to decide to aggregate a pixel to the region. Values are from the gray scale of the input image.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of aggregation or FAILURE.

Examples

Aggregates pixels to tangram pieces:

```
pbinarization 96 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pcontrastaggregation 8 20 b.pan tangram.pan out.pan
```

See also

Segmentation

C++ prototype

```
Errc PContrastAggregation( const Reg2d &rg_in, const Img2duc &im_in,
Reg2d &rg_out, int connexity, Uchar threshold );
```

Version française

Croissance des régions d’une carte selon le contraste intérieur.

Author: Régis Clouard

pcontrastmerging

Performs priority region merging based on contrast criterion.

Synopsis

pcontrastmerging *number threshold* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

pcontrastmerging merges connected regions of the input image *rg_in* if the difference between the inner contrast of the region is lower than the specified *threshold*.

Two regions are connected if there exists a link between the related nodes in the input graph *gr_in*.

The principle of the algorithm is as follows:

- For each region of the input region map *rg_in*:
- If the difference between the criterion value of the two connected regions ≤ *threshold* then merge them into one region.

The algorithm uses the priority merging that consists in merging regions with the lower difference.

The output region map *reg_out* defines the new regions and the output graph *gr_out* defines the new relationship between regions.

The contrast is calculated as follows:

```
contrast(R)= MAX(R) - MIN(R)
```

Parameters

- *number* specifies the number of allowed merging. If *number* = -1 then all possible merging are done.
- *threshold* specifies the maximum difference allowed between two regions to decide to merge them. Values are from the gray scale of the input image.

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.
- *gr_out*: a graph.

Result

Returns the number of merging.

Examples

Merges regions yielded by a quadtree splitting process:

```
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
```

See also

Segmentation

C++ prototype

```
Errc PContrastMerging ( const Reg2d &rg_in, const Graph2d &gr_in,
const Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, long number,
float threshold);
```

Version française

Fusion prioritaire de régions selon le critère du contraste.

Author: Laurent Quesnel

pcontrastquadtree

Performs quadtree (or octree) segmentation based on contrast uniformity.

Synopsis

pcontrastquadtree *threshold* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

pcontrastquadtree segments the input image *im_in* into homogeneous regions. Homogeneous regions are regions that have an inner contrast ≤ *threshold*.

The principle of the algorithm is as follows:

- At the begin consider the image as the first block.
- If the block violates the uniformity predicate (i.e. inner contrast ≤ *threshold*) then split the block into four equally sized sub-blocks and then apply the algorithm recursively on each sub-blocks.

Therefore, the result is composed of rectangular regions.

The uniformity degree is calculated from:

```
contrast(R)= MAX(R) - MIN(R).
```

For 3D image, the output region map is composed of octree regions.

Parameters

- *threshold* is the maximum contrast value to decide if a region is homogeneous or not. Values are from the gray scale of the input image *im_in* (eg., 0-255 for Uchar image).

Inputs

- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of regions.

Examples

Builds the quadtree of tangram.pan:

```
pcontrastquadtree 10 tangram.pan a.pan
```

See also

Segmentation

C++ prototype

```
Errc PContrastQuadtree( const Img2duc &im_in, Reg2d &rg_out, Uchar
threshold );
```

Version française

Segmentation d’une image par quadtree (ou octree) selon le contraste.

Author: Laurent Quesnel

pcontrastthresholding

Performs multi-thresholding on image based on the boundary contrast value.

Synopsis

pcontrastthresholding *nbclass* [*-m mask*] [*im_in*|-] [*im_amp*|-] [*im_out*|-]

Description

pcontrastthresholding classifies the input image pixels into a small number of clusters according to their value. Every pixel p of the input image is assigned to a cluster identified by the related threshold value:

```
if threshold[k-1]<im_out[p]<=threshold[k].
then im_out[p]=threshold[k]
```

The last threshold is equal to 255.

The classification is based on the analysis of the gradient magnitude along the regions boundaries given in the *im_amp* image. The principle is based on the Kohler's algorithm :
Let p and q be two neighbors. A boundary between p and q is detected by a threshold t if:

```
im_in[p]<=t<=im_in[q] or im_in[q]<=t<=im_in[p].
```

Thus, the set of boundary detected by t is:

```
K(t) = { pair(p,q) / p and q neighbors and im_in[p]<=t<=im_in[q] or im_in[q]<=t<=im_in[p] }
```

The total contrast of the boundaries detected by t is:

```
C(t) = SUM(MIN(ABS(t-im_in[p]),ABS(t-im_in[q]))
```

The mean contrast is:

```
Cm(s) = C(t) / card(K(t))
```

Then the thresholds are located as minima of the mean contrast function.

Parameters

- *nbclass* specifies the number of output clusters. It is a positive integer.

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).
- *im_amp*: a grayscale image that contains gradient magnitude values (Img2duc Img3duc).

Outputs

- *im_out*: a grayscale image of bytes (Img2duc, Img3duc).

Result

Returns the number of thresholds.

Examples

Segments tangram pieces:

```
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
```

See also

Thresholding

C++ prototype

```
Errc PContrastThresholding( const Img2duc &im_in1, Img2duc &im_in2,
Img2duc &im_out, int nbclass );
```

Version française

Multi-seuillage de l’image par analyse du contraste aux frontières.

Reference

R. Kohler, "A segmentation system based on thresholding", *CGIP*, No. 15, pp 319-338, 1981.

Author: Régis Clouard

pcontrastvalue

Measures the global contrast value of grayscale image or graph.

Synopsis

pcontrastvalue [*im_in*|-] [*col_out*|-]

Description

pcontrastvalue measures the global contrast of the input image *im_in* or the input graph *im_in*.

The contrast is the difference between the minimum and maximum values:

```
contrast(im_in) = max(im_in) - min(im_in).
```

For image, the contrast is measured from the pixel values.

For graph, the contrast is measured from the node values.

The contrast values for each band are stored in the collection *col_out*.

Inputs

- *im_in*: an image or a graph.

Outputs

- *col_out*: a collection of float values.

Result

Returns the global contrast value (for the first band only). This value can be get using operator **pstatus**.

Examples

Measures the global contrast of the tangram.pan (Unix version):

```
pcontrastvalue tangram.pan col.pan
var=‘pstatus‘
echo "Contrast = $val"
```

Measures the global contrast of the tangram.pan (MsDos version):

```
pcontrastvalue tangram.pan col.pan
call pstatus
call pset var
echo Contrast = %val%
```

See also

Image Features Extraction

C++ prototype

```
Float PContrastValue( const Img2duc &im_in, Collection & col_out );
```

Version française

Calcul du contraste global d’une image ou d’un graphe.

Author: Régis Clouard

pconvexhull

Builds convex hull of regions.

Synopsis

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

Description

pconvexhull builds the convex hull for each region of the input region map *rg_in*. The convex hull keeps the same label value as the related region.

Warning: It might occur some overlapping between convex hulls. Therefore, a convex hull can hide another.

Inputs

- *rg_in*: a region map.

Outputs

- *rg_out*: a region map.

Result

Returns SUCCESS or FAILURE.

Examples

Draws the convex hull of the regions in the region map rin.pan:

```
pconvexhull rin.pan rout.pan
```

See also

Region

C++ prototype

```
Errc PConvexHull( const Reg2d &, Reg2d &im_out );
```

Version française

Calcul de l’enveloppe convexe des régions.

Author: Régis Clouard

pconvexityselection

Selects regions from convexity degree.

Synopsis

pconvexityselection *relation threshold* [*-m mask*] [*rg_in*|*-*] [*rg_out*|*-*]

Description

pconvexityselection selects regions from their convexity degree. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

Convexity is the relative amount that a region differs from a convex region. It is calculated as follows:

```
convexity = region area / convex hull area.
```

The maximum value is 1.0 for convex region (eg. circle, square).

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is an real value from [0..1] where 1 is for convex shape.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *rg_out*: a 2D region map.

Result

Returns the number of selected regions.

Examples

Selects regions with convexity degree ≥ 0.5:

```
pconvexityselection 2 0.5 rin.pan rout.pan
```

See also

Region

C++ prototype

```
Errc PConvexitySelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, Ushort threshold );
```

Version française

Sélection de régions sur leur valeur de convexité.

Author: Régis Clouard

pconvolution

Convolves image with kernel.

Synopsis

pconvolution *filename* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pconvolution convolves an image *im_in* by the kernel given in the *filename*.

Convolution consists in:

```
im_out[y][x]= SUM {im_in[y+l][x+k]* mask[l][k]} / norm;
```

The result is normalized by the sum of each kernel coefficients (`norm`) or by 1 if the sum is null.

Note: The image border of the half size of the kernel is not affected by the convolution. It is set to 0 in the result *im_out*.

The *filename* is a text file which contains several lines formated as follows:

- the size of the kernel: number_of_plane*number_rows*number_columns;
- the coefficients (real or integer) are organized in sequence separated by blanks. The order is planes, rows and then columns.

```
nprof*nrow*ncol
c1 c2 c3 ...
```

For example, the following kernel is used to approximate the 2D laplacian filtering:

```
3*3
-1.0 -1.0 -1.0 -1.0 8 -1 -1 -1 -1
```

Inputs

- *im_in*: an image.

Outputs

- *im_out*: a Float image.

Result

Returns SUCCESS or FAILURE.

Examples

The followings kernel can be used to perform mean filter on tangram.pan:

```
kernel.txt:
3*3
1 1 1 1 1 1 1 1 1

pconvolution kernel.txt tangram.pan a.pan
```

See also

Arithmetic

C++ prototype

```
Errc PConvolution( const Img2duc &im_in1, Img2dsf &im_in2, char*
filename );
```

Version française

Convolution d’une image par un noyau.

Author: Régis Clouard

pcorrelationbinarization

Performs binarization on image using maximum correlation criterion.

Synopsis

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

Description

pcorrelationbinarization classifies pixels of the input image *im_in* into 2 clusters: the background and the foreground. The threshold value is determined as the gray level value `s` that maximizes the total amount of correlation provided by the background and the foreground separately. The total amount of correlation for threshold s is:

```
TC(s) = Cb(s) + Cf(s) { correlation for background + correlation for foreground }
      = -ln[G(s)*G'(s)] +2*ln[P(s)*(1-P(s))]
  where P(s) = SUM{i=0->s} [p(i)]
  and G(s) = SUM{i=0->s} [p(i)^2)]
  and G'(s) = SUM{i=s->m-1} [(p(i)^2]
  and pi = fi/W*H
```

The maximum correlation criterion is to determine the threshold smax such that:

```
TC(smax) = max TC(s)
```

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).

Outputs

- *im_out*: a grayscale image of bytes (Img2duc, Img3duc).

Result

Returns the threshold value.

Examples

Segments the tangram pieces:

```
pcorrelationbinarization tangram.pan a.pan
```

See also

Thresholding

C++ prototype

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

Version française

Binarisation de l’image par maximisation de la corrélation interclasse.

Reference

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.

Author: Régis Clouard

pcreatearray

Creates array in collection.

Synopsis

pcreatearray *name type size value* [*col_out*|-]

Description

pcreatearray creates a new collection *col_out* with one array named *name* that contains *size* values *value* of type *type*.

Parameters

- *name* is the base name of the array. It is string without blank character.
- *type* is a basic type among `Char, Uchar, Short, Ushort, Long, Ulong, Float, Double`.
- *size* is the size of the array.
- *value* is the initial value of type *type*.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Creates an array with 128 float values 10.5:

```
pcreatearray foo float 128 10.5 col.pan
pcol2txt col.pan
```

See also

Array

Prototypes C++

```
Errc CreateArray( Collection &col_out, Long size, Uchar value,
std::string name );
```

Version française

Création d’une collection contenant un vecteur vierge.

Author: Alexandre Duret-Lutz

pdelaunay

Builds the Delaunay graph from region map.

Synopsis

pdelaunay [*-m mask*] [*rg_in*|-] [*gr_out*|-]

Description

pdelaunay builds the Delaunay graph from the input region map *rg_in*. A Delaunay triangulation of a vertex set is a triangulation of the vertex set with the property that no vertex in the vertex set falls in the interior of the circumcircle (circle that passes through all three vertices) of any triangle in the triangulation.

Delaunay graph is the dual of the Voronoi diagram, where each element is connected to all elements that share a Voronoi edge. A Voronoï polygon associated to a site Pi is the region Vor(Pi) (each region Vor(Pi) is the set of point (x,y) nearest to the point of Vor(Pi) than any other regions) such as each point of P has Pi as the nearest site.

Seeds given in the region map *rg_in* are the site of the diagram to be built. There are the nodes of the output graph *gr_out*. **A seed is a region defined by 1 pixel and with a unique label**. If the seeds are not by a region with 1 pixel, **pdelaunay** returns an error.

Each edge weight is set with 1.0.

Inputs

- *rg_in*: a region map that contains the seeds.

Outputs

- *gr_out*: a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Calculates the Delaunay graph from the centers of mass of tangram pieces:

```
pbinarization 90 1e30 tangram.pan a.pan
plabeling 8 a.pan r1.pan
pcenterofmass r1.pan r2.pan
pdelaunay r2.pan g.pan
```

See also

Graph, pvoronoi

C++ prototype

```
Errc PDelaunay( const Reg2d &rg_in, Graph2d &gr_out );
```

Version française

Construction du graphe de Delaunay discret.

Author: Sébastien Bougleux

pdenoisePDE

Performs anisotropic smoothing on image.

Synopsis

pdenoisePDE *nb_iter amplitude sharpness anisotropy alpha sigma* [*im_in*|-] [*im_out*|-]

Description

pdenoisePDE regularizes color or multispectral images. This operator acts as an image regularizer, by doing anisotropic smoothing of the input multi-valued image. Such regularization technique is very efficient to remove local image artifact such as noise or compression artifact. It is also anisotropic and thus the smoothing preserves the important image structures such as edges, corners or discontinuities.

The processing time can be very high depending on the given parameter values.

Parameters

- *nb_iter* defines the number of iterations. The greater is the number of iterations, the stronger is the smoothing. A typical value is 2.
- *amplitude* defines the smoothing amplitude for one iteration. The greater is the amplitude, the quicker is the processing. In general, values are from 5..200.
- *sharpness* defines the edge contrast of the edges to be preserved. The greater is the value, the more there are contours. A value of 0 indicates that each pixel is smoothed with the same strength eventually in different directions. Typical values are from 0..2.
- *anisotropy* defines anisotropic degree. An anisotropic smoothing (anisotropy=1) is oriented along the edge direction. An isotropic smoothing (anisotropy=0) is not oriented. According to the noise type, it can be useful to restrict the anisotropic to avoid texture effects apparition. Allowed values are 0 or 1.
- *alpha* defines the noise scale. It represents the estimated variance of the noise. A typical value is 0.1.
- *sigma* defines the image geometry scale. Before each iteration, the local image geometry is evaluated. The sigma parameter corresponds to a pre-smoothing of this geometry. The greater is sigma, the less details are preserved but the more the smoothing is coherent. In general, a value less than 1 is convenient. A typical value is 0 8.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Denoises tangram.pan with only 2 iterations:

```
pdenoisePDE 2 100 2 0.7 0.1 0.8 tangram.pan a.pan
```

See also

Filtering

C++ prototype

```
Errc PDenoisePDE( const Imx3d &ims, Imx3d &imd, Long nb_iter, Float amplitude, Float sharpness, Float anisotropy, Float alpha, Float sigma );
```

Version française

Régularisation d’images multivaluées par lissage anisotrope basé EDP.

Inportant notice

The source code of this Pandore operator is governed by a specific Free-Software License (the CeCiLL License), also applying to the CImg Library. Please read it carefully, if you want to use this module in your own project (file CImg.h).

IN PARTICULAR, YOU ARE NOT ALLOWED TO USE THIS PANDORE MODULE IN A CLOSED-SOURCE PROPRIETARY PROJECT WITHOUT ASKING AN AUTHORIZATION TO THE CIMG LIBRARY AUTHOR (http://www.greyc.ensicaen.fr/~dtschump/)

Reference

D. Tschumperlé, "*Fast Anisotropic Smoothing of Multi-Valued Images using Curvature-Preserving PDE’s*", Cahier du GREYC No 05/01, Avril 2005.

Author: D. Tschumperlé

pdepth2graylevel

Converts depth values to gray levels.

Synopsis

pdepth2graylevel *threshold* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pdepth2graylevel builds a 2D image from a 3D image, where depths of the 3D image *im_in* are converted to gray level into the 2D image *im_out*. The depth is defined as the slice of the first pixel at the same xy-coordinates that have a value > *threshold*. The first slice is supposed to be at the depth 0.

The algorithm is as follows:

```
for (z=0; z< depth(im_in); z++)
    if (im_in[z][y][x] > threshold ) then im-out[p.y][p.x] = p.z;
```

Parameters

- *threshold* defines the maximum transparency color. Each gray level ≤ *threshold* are considered as transparent color.

Inputs

- *im_in*: a 3D gray level image of Uchar.

Outputs

- *im_out*: a 2D Slong image (Img2dsl).

Result

Returns SUCCESS or FAILURE.

Examples

Builds a Random Dot Stereogram from the 3D image cyto3d.pan:

```
pdepth2graylevel 50 cyto3d.pan i0.pan
pmultcst 10 i0.pan i1.pan
prds i1.pan rds_out.pan
```

See also

Utility, pgraylevel2depth

C++ prototype

```
Errc PDepth2Graylevel( const Img3duc &im_in, Img2dsl &im_out, long
threshold );
```

Version française

Construction d’une image de niveau de gris 2D à partir d’une image 3D.

Author: Jean-Marie Janik

pderavi

Performs multi-thresholding on image using Deravi algorithm.

Synopsis

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

Description

pderavi classifies the input image pixels into a small number of clusters according to their value. Every pixel p of the input image is assigned to a cluster identified by the related threshold value:

```
if threshold[k-1]<im_out[p]<=threshold[k].
then im_out[p]=threshold[k]
```

The last threshold is equal to 255.

The number of clusters and the value of the thresholds are determined from the Deravi algorithm. It is based on the measure of the conditional probability of transition. For each gray level i the busyness ratio is:

```
P(i) = (P1(i) + P2(i) ) / 2;

P1(i) = [SUM(SUM((Tkl)*(Tkl))] / [SUM(SUM(Tkl))+ SUM(SUM(Tpq))]
with k=[0..i], l=[i+1..N-1], p=[0..i], q=[0..i]

P2(i) = [SUM(SUM((Tkl)*(Tkl))] / [SUM(SUM(Tkl))+ SUM(SUM(Tpq))]
with k=[i+1..N-1], l=[0..i], p=[i+1..N-1], q=[i+1..N-1]
```

The co-occurrence matrix Tkl contains the number of times the gray level k is a neighbor of the gray level l considering the neighborhood Nxy={ (x,y+1), (x+1,y) }.

Then the thresholds are located as regional minima of the conditional probability of transition function. The minima are searched in the space of *length* gray levels around the gray level i.

Notice: This operator can only work on grayscale image of bytes (Img2duc, Img3duc).

Parameters

- *length* defined the length of the search space of the regional minima. It is defined in gray level unit. The greater is the length, the less there are thresholds. A typical value is 10.

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).

Outputs

- *im_out*: a grayscale image of bytes (Img2duc, Img3duc).

Result

Returns the number of thresholds.

Examples

Segments tangram.pan and displays the number of thresholds:

```
pderavi 15 tangram.pan out.pan
pstatus
```

See also

Thresholding

C++ prototype

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

Version française

Multiseuillage de l’image par analyse de la matrice de co-occurences selon Deravi.

Reference

F. Deravi et al., "Gray level thresholding using second-order statistics", *Pattern Recognition Letter*, Vol. 1, No. 5-6, pp. 417-422, 1983.

Author: Régis Clouard

pderiche

Computes gradient magnitude and maxima localization using Deriche algorithm.

Synopsis

pderiche *sigma* [*-m mask*] [*im_in*|-] [*im_mag*|-] [*im_dir*|-]

Description

pderiche computes the gradient magnitude and the gradient direction images and performs the maxima localization. The output image can then be used to locate the contours. The output image *im_out* is built with the maximum magnitude value in the direction of the gradient. Other values are set to 0.

The gradient magnitude value reflects the amount of grayscale variation in this point. The more is the variation, the greater is the value.

The gradient extraction and localization is done in three steps :

1. smoothing,
2. gradient computing,
3. local maxima extraction.

The direction is the atan(dy/dx) measured in radians. The direction image *im_dir* is necessarily of type float (values in [0..2*PI]).
Warning: the direction follows the image coordinate system, it means that it’s going clockwise when displayed, since the y coordinates are inversed in the image system.

Note: The image border of size 1 is set to 0.

Parameters

- *sigma* is a real value that controls the strength of the smoothing. Values are generally from the interval [0..10]. The lower is the *sigma* value, the stronger is the smoothing and thus the less there are contour points in the output image.

Inputs

- *im_in*: a 2D grayscale image.

Outputs

- *im_out*: the gradient magnitude image of same type as the input image.
- *im_dir*: the gradient direction image of type float.

Result

Returns SUCCESS or FAILURE.

Examples

Performs an edge detection for the tangram.pan image:

```
pderiche 1 tangram.pan a.pan b.pan
pbinarization 10 1e30 a.pan c.pan
```

See also

Edge detection

C++ prototype

```
Errc PDeriche( const Img2duc &im_in, Img2duc &im_mag, Img2duc
&im_dir, float sigma );
```

Version française

Détection et localisation des contours par l’algorithme de Deriche.

Author: Carlotti & Joguet

pderichesmoothing

Performs Deriche filtering on image.

Synopsis

pderichesmoothing *alpha* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pderichesmoothing performs Deriche smoothing of the input image *im_in*.

Parameters

- *alpha* is a real value that controls the strength of the smoothing. Values are generally from the interval [0..10]. The lower is the *alpha* value, the stronger is the smoothing. A typical value, is 1.0.

Inputs

- *im_in*: a 2D image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Performs Deriche smoothing on tangram.pan:

```
pderichesmoothing 1 tangram.pan out.pan
```

See also

Filtering

C++ prototype

```
Errc PDericheSmoothing( const Img2duc &im_in, Img2duc &im_out,
double alpha );
```

Version française

Lissage de Deriche.

Author: Carlotti & Joguet

pdif

Performs difference between images or graphs and non symmetrical difference between region maps.

Synopsis

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

Description

pdif computes the difference between the two inputs *im_in1* and *im_in2*.

If *im_in1* and *im_in2* are images then the new image *im_out* is built with the difference between each pixel:

```
pixel(im_out) = ABS(pixel(im_in1) - pixel(im_in2));
```

The two inputs must be of the same type.

For color or multispectral image, the difference is computed separately on each band.

If *im_in1* and *im_in2* are graphs then the new graph *im_out* is built with the difference between each node values.

If *im_in1* and *im_in2* are region maps **pdif** computes the symmetrical difference between region maps:

```
Union(im_in1,im_in2) - Intersection(im_in1,im_in2).
```

Inputs

- *im_in1*: an image, a graph or a region map.
- *im_in2*: an image, a graph or a region map.

Outputs

- *im_out*: an object of the same type as im_in1 and im_in2.

Result

Returns SUCCESS or FAILURE.

For region map, returns the new higher label value.

Examples

```
pdif a.pan b.pan c.pan
```

See also

Arithmetic

C++ prototype

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

Version française

Différence d’images ou de graphes et différence symétrique entre cartes de régions.

Author: Régis Clouard

pdilatation

Performs morphological dilatation.

Synopsis

pdilatation *num_se halfsize* [*-m mask*][*im_in*|-][*im_out*|-]

Description

pdilatation dilates the points of stronger contrast according to a structuring element.

Dilatation corresponds to the operation: replaces the central pixel p by the maximum of its neighbors where the neighbors are specified by the structuring element.

```
dilatation(p) = MAX(neighbors(p)).
```

The structuring element is specified by its type *num_se* and its size *halfsize*.

For a binary image, dilatation dilates white areas.

For the region maps, dilatation dilates only regions that touch the background and region with the higher label are privileged.

For the color images, the lexicographic order is used: initially by using band X, in the event of equality by using the band Y then band Z.

Parameters

- *num_se* specifies the type of the structuring element:

  case of 2D:

  - 0: diamond (4-connexity)
  - 1: square (8-connexity)
  - 2: disc

  case of 3D:

  - 0: bipyramid (6-connexity)
  - 1: cube (26-connexity)
  - 2: sphere

  (This parameter is ignored for 1D image)
- *halfsize* specifies the half-size of the structuring element. For example, a half-size of 1 for a square gives a structuring element of size 3x3.

Inputs

- *im_in*: an image (1D, 2D, 3D) or a region map.

Outputs

- *im_out*: an image (or a region map) of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Performs a Black Top Hat with small 17x17 square structuring element:

```
pinverse tangram.pan i0.pan
pdilatation 1 8 i0.pan i1.pan
perosion 1 8 i1.pan i2.pan
pdif i2.pan i0.pan out.pan
```

See also

Morphology, psedilatation, perosion

C++ prototype

```
Errc PDilatation( const Img2duc &im_in, Img2duc &im_out, int num_se,
int halfsize );
```

Version française

Dilatation des points de fort contraste d’une image.

Author: Régis Clouard

pdilatationreconstruction

Performs reconstruction by dilatation.

Synopsis

pdilatationreconstruction *connexity* [*-m mask*] [*im_in1*|-] [*im_in2*|-][*im_out*|-]

Description

pdilatationreconstruction performs a geodesic reconstruction by dilation of the markers image *im_in1* in the mask image *im_in2*.

The two images must be of the same type, and the image of markers *im_in1* must be lower or equal in intensity to the image of mask *im_in2*.

The reconstruction by dilation according to the *connexity* consists in the following operation applied until idempotence:

```
im1=MIN(im_in1, im_in2)
imdilat=dilatation(im1, connexity)
im1=MIN(imdilat, im_in2)
```

For the color images, it is the lexicographic order which is used: initially by using band X, in the event of equality by using the band Y then band Z.

Parameters

- *connexity* specifies the relationship between a pixel and its neighbors. It is an integer from: 4 or 8 for 2D or 6 or 26 for 3D.

Inputs

- *im_in1*: an image.
- *im_in2*: an image of the same type as *im_in1*.

Outputs

- *im_out*: an image of the same type as *im_in1*.

Result

Returns SUCCESS or FAILURE.

Examples

Performs white geodesic Top Hat with small 17x17 square structuring element:

```
perosion 1 8 tangram.pan i1.pan
pdilatationreconstruction 8 i1.pan tangram.pan i2.pan
pdif tangram.pan i2.pan out.pan
```

See also

Morphology, perosionreconstruction

C++ prototype

```
Errc PDilatationReconstruction( const Img2duc &im_in1, const Img2duc
&im_in2, Img2duc &im_out, int connexity );
```

Version française

Reconstruction morphologique par dilatation.

Author: Régis Clouard

pdistance

Computes euclidean distance map to nearest contours.

Synopsis

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

Description

pdistance computes the distance map to the nearest contour. The output image *im_out* is a float image where each pixel is set with the distance to the nearest contour point. A contour is a chain of connected non null pixels.

If the input image is a graph then the distance is computed from the node values.

The algorithm uses the Eikonale equation (fast marching). It corresponds to a good approximation of the euclidean distance.

Inputs

- *im_in*: a grayscale image (Img2duc or Img3duc) or a graph.

Outputs

- *im_out*: a float image (Img2dsf or Img3dsf) or a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Closed contours yielded by a simple edge detection of tangram.pan:

```
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
```

See also

Contour

C++ prototype

```
Errc PDistance( const Img2duc &im_in, Img2dsf &im_out );
```

Version française

Calcul d’une image de distance euclidienne aux contours.

Authors: Jean-Marie Janik & Abderrahim Elmoataz

pdistance1

Computes distance map to nearest contour.

Synopsis

pdistance1 *d1 d2 d3* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pdistance computes the distance map to the nearest contour. The output image *im_out* is a float image where each pixel is set with the distance to the nearest contour point. A contour is a chain of connected non null pixels.

The distance is computed from the specified distance between neighbors: *d1*, *d2* and *d3*:

```
+d2 +d1 +d2
+d1  x  +d1
+d2 +d1 +d2
```

For 3D, *d3* is the distance between the diagonal neighbors.

If the input image is a graph then distance is computed from the node values.

Parameters

- *d1*, *d2* and *d3* are real values that specify the distance metric. For example, the following metric can be used with convenient values:
  - Euclidean: d1 = 1 ; d2 = sqrt(2), d3 = sqrt(3);
  - Chessboard: d1 = 1 ; d2 = 2; d3 = 3;
  - Manhattan: d1 = 1 ; d2 = 1; d3 = 1;

Inputs

- *im_in*: a grayscale image (Img2duc or Img3duc) or a graph.

Outputs

- *im_out*: a float image (Img2dsf or Img3dsf) or a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Closed contours yielded by a simple edge detection of tangram.pan:

```
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
```

See also

Contour

C++ prototype

```
Errc PDistance1( const Img2duc &im_in, Img2dsf &im_out, float d1,
float d2, float d3 );
```

Version française

Calcul d’une image de distance quelconque aux contours.

Author: Régis Clouard

pdiv

Performs division between images or graphs.

Synopsis

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

Description

pdiv computes the division of the input *im_in1* by the input *im_in2*.

If *im_in1* and *im_in2* are images then the new image *im_out* is built with the division of each pixel. The problem of 0 is solved as follows:

```
if (pixel(im_in) == 0 )
   pixel(im_out)= 0;
else
   pixel(im_out) = pixel(im_in1)  / pixel(im_in2);
```

The two inputs must be of the same type. The output image *im_out* is a Float image.

For color or multispectral image, the division is computed separately on each band.

If *im_in1* and *im_in2* are graphs then the new graph *im_out* is built with the division of each node value.

Inputs

- *im_in1*: an image or a graph.
- *im_in2*: an image or a graph.

Outputs

- *im_out*: a Float image or a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Divides image a.pan by image b.pan:

```
pdiv a.pan b.pan c.pan
```

See also

Arithmetic

C++ prototype

```
Errc PDiv( const Img2duc &im_in1, const Img2duc &im_in2, Img2dsf
&im_out );
```

Version française

Division d’images ou de graphes.

Author: Régis Clouard

pdivcst

Diviplies constant to image, graph or region map.

Synopsis

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

Description

pdivcst builds the new output *im_out* by dividing the specified constant to each value of *im_in*.

For image, **pdivcst** divides the specified value to each pixel. The values are clipped if they are greater than the maximum possible value or lower than the minimum:

```
val = pixel(im_in) * cst;
if (val > MAX) pixel(im_out) = MAX;
else if (val < MIN) pixel(im_out) = MIN;
else pixel(im_out) = val;
```

For color or multispectral image, **pdivcst** is computed separately on each band.

For region map, **pdivcst** divides the specified value to each label.

For graph, **pdivcst** divides the specified value to each node value.

The output file is of the same type as the input file.

Parameters

- *cst* is a real value.

Inputs

- *im_in*: an image, a graph or a region map.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

For region map, returns the new higher label value.

Examples

Divides the tangram.pan pixel values by 2:

```
pdivcst 2 tangram.pan a.pan
```

See also

Arithmetic

C++ prototype

```
Errc PDivCst( const Img2duc &im_in, Img2duc &im_out, float cst );
```

Version française

Division par une constante des valeurs d’une image, d’un graphe ou d’une carte de région.

Author: Régis Clouard

pdivneumann

Computes the divergence by backward finite differences.

Synopsis

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

Description

pdivneumann computes the divergence by backward finite differences. The result is a grayscale image *im_out*, where:

```
 im_out(i,j) = (im_in1(i,j)-im_in1(i-1,j)) + (im_in2(i,j)-im2(i,j-1)),
```

with special care (Neumann) at boundaries:

```
im_in1(1,j) and -im_in1(n-1,j)
im_in2(i,1) and -im_in2(i,n-1)
```

Inputs

- *im_in1*: a 2D image.
- *im_in2*: a 2D image (same type as im_in1).

Outputs

- *im_out*: an image of the same type as im_in1.

Result

Returns SUCCESS or FAILURE.

Examples

Implements the gradient and divergence operators with Neumann boundary conditions such that one is the adjoint of the other, i.e. <grad x,u> = <-div u,x>. The script checks this identity.

```
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
```

See also

Edge detection, pgradneumann

C++ prototype

```
Errc PDivNeumann( const Img2d<U> &im_in1, Img2d<U> &im_in2, Img2d<U>
&im_out );
```

Version française

Calcul de la divergence par différence finies décentr&eacutee à gauche.

Author: Jalal Fadili

pdivval

Divides image bands with constants stored in collection.

Synopsis

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

Description

pdivval builds the new output *im_out* by dividing each band of the input image *im_in* with constants stored in the collection *col_in*. The first bands is divided with the first constant in the collection, the second bdans with the second constant, etc.

The values are clipped if they are greater than the maximum allowed value or lower than the minimum:

```
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;
```

The output file is of the same type as the input file.

Inputs

- *col_in*: a collection with a number of float values equals to the number of bands of the input image.
- *im_in*: an image.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Divides tangram.pan by its mean value:

```
pmeanvalue tangram.pan col.pan
pdivval col.pan tangram.pan a.pan
```

More examples

See also

Arithmetic

C++ prototype

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

Version française

Division d’une image par des constantes stockÃ©es dans une collection.

Author: Régis Clouard

pdraw

Draws by hands onto Pandore object.

Synopsis

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

Description

pdraw is a graphical image manipulator for Pandore objects. Available Pandore objects are:

- images;
- regions map;
- graphs.

The output object *im_out* is an unsigned char grayscale image where lines are represented by connected pixels with value 255.

The Qt version of **pdraw** also accepts Qt options at the beginning of the arguments list:

```
pvisu -style motif tangram.pan a.pan
```

All tools are accessible through menu or by shortcut key.

Inputs

- *im_in*: an image, a region map or a graph.

Inputs

- *im_out*: a byte image (Uchar).

Result

Returns the process ID (PID) of the related process or FAILURE.

Examples

Creates the image a.pan by drawing on the image tangram.pan and labels regions within closed contours:

```
pdraw tangram.pan a.pan
pboundarylabeling a.pan b.pan
pvisu b.pan
```

See also

Visualization, pvisu.

Version française

Interface de dessin sur une image.

Author: Régis Clouard

pdwt

Performs Direct Wavelet Transform.

Synopsis

pdwt *scale* [*im_in*|-] [*col_in*|-] [*im_out*| -]

Description

pdwt calculates the wavelet coefficients of the input image *im_in1* according to the pyramidal algorithm. For example, with *scale*=1, there are 4 subimages:

```
[1][2]
[3][4]
```

where [1] is the approximation image with undersampling factor of 2. and [2], [3], [4] correspond to the signal detail along each privileged direction (resp. horizontal, vertical, diagonal) undersampled with a factor 2.

The filter coefficients are stored from the input collection *col_in* and can be generated from the operator pqmf.

Important: The operator works with any image size. However, the algorithm needs image with a size power of 2. Therefore, the input image is magnified so as the size is closest to power 2. So output image has necessarily a size power of 2.

Parameters

- *scale* specifies the number of levels to be used for the decomposition of the input image.

Inputs

- *im_in*: a 2D float image.
- *col_in*: a collection that contains the filter coefficients.

Outputs

- *im_out*: a 2D float image.

Result

Returns SUCCESS or FAILURE.

Examples

Builds a synthetic image (a square) to illustrate the Gibbs phenomenon in wavelets analysis.

```
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
```

See also

Frequency, pidwt, pqmf

C++ prototype

```
Errc PDwt( const Img2duc &im_in, const Collection &col_in, Img2dsf
&im_out, int scale );
```

Version française

Calcul de la transformée en ondelettes dyadiques biorthogonales d’une image.

Author: Ludovic Soltys

peccentricityselection

Selects regions from eccentricity degree.

Synopsis

peccentricityselection *relation threshold* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

peccentricityselection selects regions from their eccentricity degree. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

Eccentricity measures how much the region deviates from being circular. It is defined as the ratio of the length of the short axis to the length of the long axis:

```
              (Mxx+Myy-sqrt( (Mxx-Myy)*(Mxx-Myy)+4*Mxy*Mxy))
eccentricity= ----------------------------------------------
              (Mxx+Myy+sqrt( (Mxx-Myy)*(Mxx-Myy)+4*Mxy*Mxy))
```

The maximum value is 1.0 for a square or a circle.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is a real value from [0..1].

Inputs

- *rg_in*: a 2D region map.

Outputs

- *rg_out*: a 2D region map.

Result

Returns the number of selected regions.

Examples

Selects regions with the highest eccentricity degree:

```
peccentricity 3 0 rin.pan rout.pan
```

See also

Region

C++ prototype

```
Errc PEccentricitySelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, Ulong threshold );
```

Version française

Sélection de régions sur leur valeur d’exentricité.

Author: Régis Clouard

pedgebasedragpruning

Cut adjacency link between regions that are separated by an edge.

Synopsis

pedgebasedragpruning [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*gr_out*|-]

Description

pedgebasedragpruning cuts adjacency link in the gr_in between two regions that have at least one edge point on their boundary line given in the *im_in*. (A n edge point is a pixel in *im_in* with a valeur >0.)

This operator prevents a merge process to merge two regions that are separted by an edge.

Input

- *rg_in*: a region map.
- *gr_in*: a graph.
- *im_in*: an edge map.

Ouput

- *gr_out*: a graph.

Result

Returns the number of cuts.

Examples

Performs a bottom-up merging processus (without and with edges):

```
pbinarization 112 255 tangram.pan a.pan
plabeling 8 a.pan b.pan
paddcst 1 b.pan c.pan
prg2gr c.pan d.pan

# without edges.
psetcst 0 tangram.pan e.pan
pmeanmerging -1 60 c.pan d.pan e.pan f1.pan g1.pan

#with edges.
```

```
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
```

See Also

Segmentation

C++ Prototype

```
Errc PEdgeBasedRAGPruning( const Reg3d &rg_in, const Graph3d &gr_in,
const Img3duc &im_in, Graph3d &gr_out);
```

Author: Régis Clouard

pedgeclosing

Performs edge closing from gradient.

Synopsis

pedgeclosing *angle length* [*-m mask*] [*im_in*|-] [*im_grad*|-] [*im_out*|-]

Description

pedgeclosing tries to close open contours of the input image *im_in* by tracking contours along the gradient magnitude values given in the input image *im_grad*.

From the end points, the tracking is done in the direction given by the maximum gradient value limited to the directions specified by the parameter *angle*. If the maximum gradient value is null then the tracking is stopped. If the tracking is longer than *length*, the tracking is stopped.

Warning: This operator need end points with only 1 neigbor. Thius, it might be necessary to use the operator ppostthinning to guaranty 1 pixel thickness.

Parameters

- *angle* specifies the angle of the tracking. Is is an integer from [0..2]:
  - If *angle*=0 then the tracking is done only in the same direction as the end point (0 degree of freedom).
  - *angle*=1 corresponds to 0, 45 and -45 degrees.
  - *angle*=2 corresponds to 0, 45, 90, -45, -90 degrees.
- *length* specifies the maximum distance to the tracking.

Inputs

- *im_in*: a 2D grayscale unsigned char image (Img2duc) which contains the contours.
- *im_grad*: a 2D grayscale image which contains the gradient magnitude values.

Outputs

- *im_out*: a 2D grayscale image.

Result

Returns the number of contours or FAILURE.

Examples

Closed contours yielded by a simple edge detection of tangram.pan:

```
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
```

See also

Contour

C++ prototype

```
Errc PEdgeClosing( const Img2duc &im_in, const Img2duc &ima, Img2duc
&im_out, Ushort vois, int angle, int length );
```

Version française

Fermeture de contours par poursuite du gradient.

Author: Régis Clouard

pedgecutting

Perform edge cutting.

Synopsis

pedgecutting *low high* [*-m mask*] [*gr_in*|-] [*gr_out*|-]

Description

pedgecutting cuts edge with a value greater or equal to *low* and lower or equal to *high*.

If *high* is lower to *low* then edge is cut if its value is lower to *high* or greater to *low*.

Paramètres

low and *high* specify the bounds of the cutting.

If *high* is lower to *low* then edge is cut if its value is lower to *high* or greater to *low*.

Input

- *gr_in*: a graph.

Ouput

- *gr_out*: a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Keeps the edge with values between 1 and 2:

```
pedgecutting 2 1 g1.pan g2.pan
```

See Also

Graph

C++ Prototype

```
Errc PEdgeCutting( const Graph &gr_in, Graph &gr_out, float low,
float high );
```

Author: François Angot

pedgevisu

Visualization of graph edge weights.

Synopsis

pedgevisu [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_out*|-]

Description

pedgevisu allows the visualization of the edges between nodes. Each boundary of the region map *rg_in* is drawn by using the value of the related edge value in the input graph *gr_in*.

Inputs

- *gr_in*: a graph.

Outputs

- *gr_out*: an image.

Result

Returns SUCCESS or FAILURE.

Examples

```
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
```

See also

Graph

C++ prototype

```
Errc PEdgeVisu( const Reg2d &rg_in, const Graph2d &gr_in, Img2dsl
&im_out );
```

Version française

Visualisation des poids des arêtes d’un graphe dans une image.

Author: François Angot

pellipsoidalapproximation

Performs ellipsoidal approximation of closed contour.

Synopsis

pellipsoidalapproximation *mode* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pellipsoidalapproximation approximates a given set of points by an ellipsoid. The points are non null pixels relying on a null background.

The parameter *mode* specifies whether the approximation is done on all points so the result is only one ellipsoid, or on each contours so the result is as many ellipsoids as there are contours. A contour is a chain of connected non null pixels.

The output image *im_out* contains a set of points that define one or several ellipsoids.

Parameters

- *mode* is an integer form [0,1] which specifies:
  - mode=0: the ellipsoid is calculated from all non null pixels.
  - mode=1: the ellipsoid is calculated on each contour chain.

Inputs

- *im_in*: a grayscale 2D image which contains the contour points.

Outputs

- *im_out*: a grayscale 2D image.

Result

Returns SUCCESS or FAILURE.

Examples

Approximates each tangram piece by an ellipsoid:

```
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
```

See also

Contour

C++ prototype

```
Errc PEllipsoidalApproximation( const Img2duc &im_in, Img2duc
&im_out, int mode );
```

Version française

Approximation ellipsoïdale d’un ensemble de points ou des contours d’une image.

Author: Julien Robiaille

pelongationselection

Selects regions from elongation factor.

Synopsis

pelongationselection *relation threshold* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

pelongationselection selects regions from their elongation factor. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

Elongation is the ratio between the length and the width of the bounding box:

```
elongation = width(bounding box)/width(bounding box).
```

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is a real value from [0..1] where 1.0 is for a square or a disc.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *rg_out*: a 2D region map.

Result

Returns the number of selected regions.

Examples

Selects the most elongated regions:

```
pelongationselection 3 0 rin.pan rout.pan
```

See also

Region

C++ prototype

```
Errc PElongationSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, float threshold );
```

Version française

Sélection de régions sur leur valeur d’élongation.

Author: Régis Clouard

penergyselection

Selects regions from energy value.

Synopsis

penergyselection *relation threshold* [*-m mask*] [*rg_in*|-] [*im_in*|-][*rg_out*|-]

Description

penergyselection selects regions from their energy value. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

Energy is measure as follows:

```
energy = SUM { im_in[p] * im_in[p]} / N
```

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is a float value.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of selected regions.

Examples

Selects regions with the highest energy:

```
penergyselection 3 0 rin.pan a.pan rout.pan
```

See also

Region

C++ prototype

```
Errc PEnergySelection(const Reg2d &rg_in, Img2duc &im_in, Reg2d
&rg_out, int relation, float threshold );
```

Version française

Sélection de régions sur leur valeur d’énergie intérieure.

Author: Régis Clouard

penergyvalue

Measures global energy of grayscale image or graph.

Synopsis

penergyvalue [*im_in*|-] [*col_out*|-]

Description

penergyvalue measures the global energy of the input image *im_in* or the input graph *im_in*.

The energy is the sum is calculated as follows:

```
energy = SUM{ im_in[p] * im_in[p]} / N
```

For image, the energy is measured from the pixel values.

For graph, the energy is measured from the node values.

The energy values for each band are stored in the collection *col_out*.

Inputs

- *im_in*: an image or a graph.

Outputs

- *col_out*: a collection of float values.

Result

Returns the global energy value (for the first badn only). This value can be get using operator **pstatus**.

Examples

Measures the global energy of the tangram.pan (Unix version):

```
penergyvalue tangram.pan col.pan
val=‘pstatus‘
echo "Energy = $val"
```

Measures the global energy of the tangram.pan (MsDos version):

```
penergyvalue tangram.pan col.pan
call pstatus
call pset val
echo Energy = %val%
```

See also

Image Features Extraction

C++ prototype

```
Float PEnergyValue( const Img2duc &im_in, Collection & col_out );
```

Version française

Calcul de l’énergie d’une image ou d’un graphe.

Author: Régis Clouard

pentropybinarization

Performs binarization on image using maximum entropy criterion.

Synopsis

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

Description

pentropybinarization classifies pixels of the input image *im_in* into 2 clusters. The threshold value is determined as the gray level value s that maximizes the total amount of information provided by the background and the foreground separately. Since information is measured by entropy, the total amount of information for threshold s is:

```
TE(s) = Eb(s) + Ef(s) { entropy for background + entropy for foreground }
      = ln[P(s)(1-P(s))] - H(s)/P(s) - H'(s)/(1-P(s))
  where P(s) = SUM{i=0->s} [p(i)]
  and H(s) = SUM{i=0->s} [p(i)*ln(pi)]
  and H'(s) = SUM{i=s->m-1} [(p(i)*ln(i)]
  and W*H is the number of pixels
  and m is the number of gray levels.
  and pi = fi/W*H
```

The maximum entropy criterion is to determine the threshold smax such that:

```
TE(smax) = max TE(s)
```

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).

Outputs

- *im_out*: a grayscale image of bytes (Img2duc, Img3duc).

Result

Returns the threshold value.

Examples

Segments the tangram pieces:

```
pentropybinarization tangram.pan a.pan
```

See also

Thresholding

C++ Prototype

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

Version française

Binarisation de l’image par maximisation de l’entropie interclasse.

Reference

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.

Author: Régis Clouard

pentropymerging

Performs priority region merging based on entropy criterion.

Synopsis

pentropymerging *number threshold* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

pentropymerging merges connected regions of the input image *rg_in* if the difference between the entropy of the region is lower than the specified *threshold*.

Two regions are connected if there exists a link between the related nodes in the input graph *gr_in*.

The principle of the algorithm is as follows:

- For each region of the input region map *rg_in*:
- If the difference between the criterion value of the two connected regions ≤ *threshold* then merge them into one region.

The algorithm uses the priority merging that consists in merging regions with the lower difference.

The output region map *reg_out* defines the new regions and the output graph *gr_out* defines the new relationship between regions.

The entropy for a region is calculated as follows:

$$entropy = - SUM \{ Pi * \log_2(Pi) \}$$

where Pi is the probability of pixel i. (Pi is computed from the normalized histogram.)

Parameters

- *number* specifies the number of allowed merging. If *number* = -1 then all possible merging are done.
- *threshold* specifies the maximum difference allowed between two regions to decide to merge them. A typical value is 0.

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.
- *gr_out*: a graph.

Result

Returns the number of merging.

Examples

Merges regions yielded by a quadtree splitting process:

```
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
```

See also

Segmentation

C++ Prototype

```
Errc PEntropyMerging( const Reg2d &rg_in, const Graph2d &gr_in,
const Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, long number,
float threshold );
```

Version française

Fusion prioritaire de régions selon le critère de l’entropie.

Author: Laurent Quesnel

pentropyquadtree

Performs quadtree segmentation based on entropy uniformity.

Synopsis

pentropyquadtree *threshold* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

pentropyquadtree segments the input image *im_in* into homogeneous regions. Homogeneous regions are regions that have an inner entropy degree ≤ *threshold*.

The principle of the algorithm is as follows:

- At the begin consider the image as the first block.
- If the block violates the uniformity predicate (i.e. inner entropy ≤ *threshold*) then split the block into four equally sized sub-blocks and then apply the algorithm recursively on each sub-blocks.

Therefore, the result is composed of rectangular regions.

The entropy in the sense of Shannon is the quantity of information held by the input image. The more seldom a pixel value is, the more information it holds and the greater the entropy is.

The entropy for a region is calculated as follows:

$$entropy = - SUM \{ Pi * \log_2(Pi) \}$$

where Pi is the probability of pixel i. (Pi is computed from the normalized histogram.)

Notice: This operator cannot worked on float image since related probabilities are not significant.

For 3D image, the output region map is composed of octree regions.

Parameters

- *threshold* is the maximum entropy value to decide if a region is homogeneous or not. Values are between 0 and the maximum value approximated by ln(N)/ln(2) (where N is the number of pixels).

Inputs

- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of regions.

Examples

Builds the quadtree of tangram.pan:

```
pentropyquadtree 4 tangram.pan a.pan
```

See also

Segmentation

C++ prototype

```
Errc PEntropyQuadtree( const Img2duc &im_in, Reg2d &rg_out, float threshold );
```

Version française

Segmentation d’une image par quadtree (ou octree) selon l’entropie.

Author: Laurent Quesnel

pentropythresholding

Performs multi-thresholding on image based on the entropy value.

Synopsis

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

Description

pentropythresholding classifies the input image pixels into a small number of clusters according to their value. Every pixel p of the input image is assigned to a cluster identified by the related threshold value:

```
if threshold[k-1]<im_out[p]<=threshold[k].
then im_out[p]=threshold[k]
```

The last threshold is equal to 255.

The number of clusters and the value of the thresholds are determined from the measure of the entropy. For each gray level i the entropy value is:

```
Entropy(k) = - SUM(Tkl * Log (Tkl)) with l in [0..k].
```

The co-occurrence matrix Tkl contains the number of times the central pixel has the gray level k and the mean of its 8 neighbors is l.

Then the thresholds are located as regional maxima of the entropy function. The maxima are searched in the space of *length* gray levels around the gray level i.

Notice: This operator can only work on grayscale image of bytes.

Parameters

- *length* defined the length of the search space of the local maxima. The greater is the length, the less there are thresholds. A typical value is 10.

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).

Outputs

- *im_out*: a grayscale image of bytes (Img2duc, Img3duc).

Result

Returns the number of thresholds.

Examples

Segments tangram.pan and displays the number of thresholds:

```
pentropythresholding 10 tangram.pan out.pan
pstatus
```

See also

Thresholding

C++ Prototype

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

Version française

Multiseuillage de l’image par analyse de l’entropie des régions.

Reference

C. Fernandez-Maloigne, "*Segmentation et caractérisation d’images de textures à l’aide d’informations statistiques*", PhD Thesis, University of Compiegne, 1989.

Author: Régis Clouard

pentropyvalue

Measures global entropy of grayscale image or graph.

Synopsis

pentropyvalue [*im_in*|-] [*col_out*|-]

Description

pentropyvalue measures the global entropy of the input grayscale image or graph *im_in*. The entropy in the sense of Shannon is the quantity of information held by the input image. The more seldom a pixel value is, the more it holds information and the greater is the entropy.

The entropy is calculated as follows:

$$entropy = - SUM \{ Pi * \log_2(Pi) \}$$

where Pi is the probability of pixel i. (Pi is computed from the normalized histogram.)

The entropy values for each band are stored in the collection *col_out*.

For image, the entropy is measured from the pixel values.

For graph, the entropy is measured from the node values.

The energy values for each band are stored in the collection *col_out*.

Inputs

- *im_in*: an image or a graph.

Outputs

- *col_out*: a collection of float values.

Result

Returns the global entropy value (for the first band only). This value can be get using operator **pstatus**.

Examples

Measures the global entropy of the tangram.pan (Unix version):

```
pentropyvalue tangram.pan col.pan
var=‘pstatus‘
echo "Entropy = $val"
```

Measures the global entropy of the tangram.pan (MsDos version):

```
pentropyvalue tangram.pan col.pan
call pstatus
call pset var
echo Entropy = %val%
```

See also

Image Features Extraction

C++ Prototype

```
Float PEntropyValue( const Img2duc &im_in, Collection & col_out);
```

Version française

Calcul de l’entropie d’une image ou d’un graphe.

Author: Régis Clouard

perosion

Perfoms morphological erosion.

Synopsis

perosion *num_se halfsize* [*-m mask*][*im_in*|-][*im_out*|-]

Description

perosion erodes points of stronger contrast according to a structuring element.

Erosion corresponds to the operation: replaces the central pixel p by the minimum of its neighbors where the neighbors are specified by the structuring element.

```
erosion(p) = MIN(neighbors(p)).
```

The structuring element is specified by its number *num_se* and its *halfsize*.

For a binary image, erosion erodes white areas.

For the region maps, erosion adds pixels with label=0 (background) at the points of erosion.

For the color images, the lexicographic is used: initially by using band X, in the event of equality by using the band Y then band Z.

Parameters

- *num_se* specifies the type of the structuring element:

  case of 2D:

  - 0: diamond (4-connexity)
  - 1: square (8-connexity)
  - 2: disc

  case of 3D:

  - 0: bipyramid (6-connexity)
  - 1: cube (26-connexity)
  - 2: sphere

  (This parameter is ignored for 1D image)
- *halfsize* specifies the half-size of the structuring element. For example, a half-size of 1 for a square gives a structuring element of size 3x3.

Inputs

- *im_in*: an image (1D, 2D, 3D) or a region map.

Outputs

- *im_out*: an image (or a region map) of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Performs a White Top Hat with small 17x17 square structuring element:

```
perosion 1 8 tangram.pan i1.pan
pdilatation 1 i1.pan i2.pan
pdif i2.pan tangram.pan out.pan
```

See also

Morphology, pseerosion, pdilatation

C++ prototype

```
Errc PErosion( const Img2duc &im_in, Img2duc &im_out, int num_es,
int halfsize );
```

Version française

Erosion des points de fort contraste d’une image.

Author: Régis Clouard

perosionreconstruction

Performs reconstruction by erosion.

Synopsis

perosionreconstruction *connexity* [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

perosionreconstruction performs a geodesic reconstruction by erosion of the markers image *im_in1* in the mask image *im_in2*.

The two images must be of the same type, and the image of markers *im_in1* must be higher or equal in intensity to the image of mask *im_in2*.

The reconstruction by erosion according to the *connexity* consists in the following operation applied until idempotence:

```
im1=MAX(im_in1, im_in2)
imerod=erosion(im1, connexity)
im1=MAX(imerod, im_in2)
```

For the image scolor, the lexicographic order is used: initially by using band X, in the event of equality by using the band Y then band Z.

Parameters

- *connexity* specifies the relationship between a pixel and its neighbors. It is an integer from: 4 or 8 in 2D or 6 or 26 in 3D.

Inputs

- *im_in1*: an image.
- *im_in2*: an image of the same type as *im_in1*.

Outputs

- *im_out*: an image of the same type as *im_in1*.

Result

Returns SUCCESS or FAILURE.

Examples

Fills hole in regions yields by a simple segmentation process of the 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
```

See also

Morphology, pdilatationreconstruction

C++ prototype

```
Errc PErosionReconstruction( const Img2duc &im_in1, const Img2duc
&im_in2, Img2duc &im_out, int connexity );
```

Version française

Reconstruction morphologique par érosion.

Author: Régis Clouard

peulernumberselection

Selects regions from Euler number.

Synopsis

peulernumberselection *relation threshold* [*-m mask*] [*rg_in*|*-*] [*rg_out*|*-*]

Description

peulernumberselection selects regions from their Euler number. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

The Euler number E for a region is defined as 1 - the amount of holes H in the region. Conversely, the amount of holes H can be determined from the Euler number: H = 1 - E.

The algorithm used to calculated the Euler number uses the local operation:
Let X(R) the number of the following 2x2 patterns (r region label for region R, et 0 other labels):

```
0 0
0 r
```

Let V(R) the number of the following 2x2 pattern

```
0 r
r r
```

then Euler(R) = X(R) - V(R)

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is an integer. It corresponds to a Euler number.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *rg_out*: a 2D region map.

Result

Returns the number of selected regions.

Examples

Selects regions with at least 2 holes (E=1-2=-1):

```
peulerselection 2 -1 rin.pan rout.pan
```

See also

Region

C++ prototype

```
Errc PEulerNUmberSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, long threshold );
```

Version française

Sélection de régions sur leur valeur de nombre d’Euler.

Author: Régis Clouard

pexp

Computes exponential of image or graph.

Synopsis

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

Description

pexp computes the exponential of the input *im_in*.

If *im_in* is an image then the new image *im_out* is built with the exponential of each pixel:

```
pixel(im_out)=exp(pixel(im_in))
```

The output image is a always a Float image.

For color or multispectral image, the exponential is computed separately on each band.

If *im_in* is a graph then the new graph *im_out* is built with the exponential of each node value.

Inputs

- *im_in*: an image or a graph.

Outputs

- *im_out*: a Float image or a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Computes the exponential of the image tangram.pan :

```
pexp tangram.pan a.pan
```

See also

Arithmetic

C++ prototype

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

Version française

Exponentiel d’une image ou d’un graphe.

Author: Régis Clouard

pexponentialfiltering

Performs exponential filtering on image.

Synopsis

pexponentialfiltering *alpha* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pexponentialfiltering applies an exponential filter to the input image *im_in*. The exponential filter is built as follows for one image row, first from left to right (h1) and then from right to left (h2):

```
h1[x]= alpha*(im_in[y][x]-h1[x-1]) + h1[x-1]
h2[x]= alpha*(h1[x]-h2[x+1]) + h[x+1]
```

The filter is then applied on each rows and each columns.

Parameters

- *alpha* is a real value from the interval [0..1]. It specifies the strength of the filtering:
  - 1: light filtering.
  - 0: strong filtering.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Performs an edge detection using the DOG algorithm (Difference of Gaussian):

```
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
```

See also

Filtering

C++ prototype

```
Errc PExponentialFiltering( const Img2duc &im_in, Img2duc &im_out,
float alpha );
```

Version française

Lissage par une exponentielle symétrique.

Author: Régis Clouard

pextractsubimage

Extracts subimage from image.

Synopsis

pextractsubimage *x y z l h p* [*im_in*|-] [*im_out*|-]

Description

pextractsubimage builds a new image *im_out* with pixels of the input image *im_in* included in the window beginning at coordinates (*x*,*y*,*z*) and with the dimension (*l*,*h*,*p*).

For region map, there is no relabeling. The regions in *im_out* keep the same label than in *im_in* even if some regions have disappeared.

Parameters

- *x*,*y*,*z* specify the coordinates in *im_in1* of the image *im_out*.
- *l*,*h*,*p* specify the dimensions of the image *im_out*. If one of the dimension is greater than the maximum then the maximum size is used.
  In case of 2D image, *z* and *p* parameters are ignored but must be given.

Inputs

- *im_in*: an image or a region map.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

For image returns SUCCESS or FAILURE. For region map returns the maximum label or FAILURE.

Examples

Extracts part of the tangram.pan from coordinates 10,20 and size 246,236 (if tangram.pan is 256x256 image).

```
pextractsubimage 10 10 0 1000 1000 0 tangram.pan a.pan
```

See also

Utility, pinsertsubimage

C++ prototype

```
Errc PExtractSubImage( const Img2duc &im_in, Img2duc &im_out, Long
cz, Long cy, Long cx );
```

Version française

Extraction d’une sous-image d’une image.

Author: Régis Clouard

pextremumsharpening

Performs constrast sharpening using extremum values.

Synopsis

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

Description

pextremumsharpening performs a contrast sharpening of the input image *im_in*. The objective is to highlight fine detail and to enhance details that are blurred. Sharpening consists in shrinking the width of intensity variation without affecting the mean intensity of regions on both sides of the variation.

The algorithm consists in replacing a pixel by the closest neighborhood minimum or maximum values.

Let W by a neighborhood, and im_in(p) a pixel of the input image:

```
if (im_in(p)-min(W) < max-im_in(p))
then im_out[p]=min
else im_out[p]=max
```

For color and multispectral images, the transform uses the marginal approach: it is applied on each band individually.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image with the same properties as im_in.

Result

Returns SUCCESS or FAILURE in case of memory faults.

Examples

Sharpens the tangram.pan image.

```
pextremumsharpening tangram.pan a.pan
```

See also

Lut transform

C++ prototype

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

Version française

Rehaussement du contraste par utilisation des valeurs extrémales.

Author: Régis Clouard

pfft

Performs Fast Fourier Transform.

Synopsis

pfft [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out1*|-] [*im_out2*|-]

Description

pfft computes the Fast Fourier Transform of the input complex image *im_in1*. The input complex image is composed of two images:

- *im_in1* is the real part;
- *im_in2* is the imaginary part.

The imaginary image must be at least empty (see pnewimage, psetcst).

The output complex images is also composed of two images:

- *im_out1* is the real part of the transform;
- *im_out2* is the imaginary part of the transform.

Fast Fourier Transform is a way of going from the spatial domain to the frequency domain:

- The spatial domain is the domain where each value at the coordinate (x,y) contains the intensity value of the related point (x',y') in the scene. The distance between 2 pixels corresponds to a real distance in the scene.

- The frequency domain is the domain where each value of the image at the coordinate F(u,v) contains a quantity such that the intensity values in the input image varies on a distance related to F. For example, suppose that the value at the related coordinate for frequency 0.1 is 20 (ie. 1 period every 10 pixels). It means that in the spatial domain of the related image, the intensity value varies from dark to clear on a distance of 10 pixels and the contrast between the dark and the clear values occupies 40 gray levels (2*20).

The Fourier transform of an image represents the likeness degree between the image seen as function f and the functions are sine and cosine with various frequencies. Each point represents a particular frequency in the spatial domain.

```
if N is the number of pixels.
F(u,v) = 1/(N*N) * Sigma(x){Sigma(y){ I(x,y)*exp(-i2PI((u*i)/N+(v*i)/N))}}
```

This equation can be interpreted as follows:
The value of the point (u,v) results from the multiplication of the spatial image with the various basis function. The basis functions are sine and cosine with increasing frequencies. F(0,0) represents the

mean intensity of the image, whereas F(N-1,N-1) represents the higher frequency.

The size of the output images *im_out1* and *im_out1* is the same as the input images *im_in1* and *im_in2*.

Inputs

- *im_in1*: a gray level image (the real part of the transform).
- *im_in2*: a gray level image with the same properties than *im_in1* (the imaginary part of the transform).

Outputs

- *im_out1*: a gray level image (the real part of the transform).
- *im_out2*: a gray level image (the imaginary part of the transform).

Result

Returns SUCCESS or FAILURE.

Examples

Computes the magnitude of the Fast Fourier Transform of tangram.pan. The imaginary part (i1.pan) is null. (Use log transform dynamic in pvisu to display 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
```

See also

Frequency, pifft, pfftshift

C++ prototype

```
Errc PFFT( const Img2duc &im_in1, const Img2duc &im_in2, Img2dsf
&im_out1, Img2dsf &im_out2 );
```

Version française

Calcul de la Transformée de Fourier Rapide d’une image.

Author: Herissay & Berthet

pfftconvolution

Performs convolution of image by kernel.

Synopsis

pfftconvolution [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

pfftconvolution performs a convolution the input image *im_in1* with the kernel given in the input image *im_in2*. The size of the kernel image *im_in2* must be lower or equal than the input image *im_in1*.

The convolution uses the frequency domain. (It differs from the pconvolution operator that uses the spatial domain.) It consists in a complex multiplication of the Fourier transform of the two input images:

- imi=fft(im_in) * fft(im_in2)
- im_out=iift(imi);

where * is the complex multiplication.

Inputs

- *im_in1*: a 2D image.
- *im_in2*: a 2D image (same type as *im_in1* and a size lower or equal to *im_in1*).

Outputs

- *im_out*: a float image of the same size as the input image *im_in1*.

Result

Returns SUCCESS or FAILURE.

Examples

Adds a motion blur on the tangram.pan The motion blur is generated by the way of an oblique line:

```
pshapedesign 10 10 0 3 10 1 line.pan
protation 0 45 line.pan line1.pan
pfftconvolution tangram.pan line1.pan out.pan
```

See also

Frequency, pconvolution, pftt, pifft

Prototype C++

```
Errc FFTConvolution( const mg2duc &im_in1, const Img2duc &im_in2,
Img2dsf &im_out );
```

Version française

Convolution d’une image par un noyau.

Author: Régis Clouard

pfftcorrelation

Performs correlation between two images.

Synopsis

pfftcorrelation [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

pfftcorrelation performs a correlation between the two input images *im_in1* and *im_in2*. The correlation determines the degree of likeness between the two images. The first input images *im_in1* size must be greater or equal than the second input image *im_in2* size.

The correlation uses the frequency domain. It consists of a complex multiplication of the Fourier transform of the first input images and the complex conjugate of the second input image *im_in2*:

- imi=fft(im_in) * conj(fft(im_in2))
- im_out=iift(imi);

where * is the complex multiplication and conj(im) is the complex conjugate of im.

Inputs

- *im_in1*: a 2D image.
- *im_in2*: a 2D image (same type and size as im_in1).

Outputs

- *im_out*: a float image.

Result

Returns SUCCESS or FAILURE.

Examples

Detects presence of tangram pieces in the tangram.pan image:

```
pextractsubimage 111 6 0 35 32 0 tangram.pan a.pan
pfftcorrelation tangram.pan a.pan b.pan
plocalmaxima 8 b.pan out.pan
```

See also

Frequency, pftt, piftt

Prototype C++

```
Errc FFTCorrelation( const Img2duc &im_in1, copnst Img2duc &im_in2,
Img2dsf &im_out );
```

English version

Corrélation de deux images.

Author: Régis Clouard

pfftdeconvolution

Performs deconvolution of image by kernel.

Synopsis

pfftdeconvolution [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

pfftdeconvolution performs a deconvolution the input image *im_in1* with the kernel given in the input image *im_in2*. The size of the kernel image *im_in2* must be lower or equal than the input image *im_in1*.

The deconvolution uses the frequency domain. It consists in a complex division of the Fourier transform of the two input images:

- imi=fft(im_in) / fft(im_in2)
- im_out=iift(imi);

where / is the complex division.

Inputs

- *im_in1*: a 2D image.
- *im_in2*: a 2D image (same type as *im_in1* and a size lower or equal to *im_in1*).

Outputs

- *im_out*: a float image of the same size as the input image *im_in1*.

Result

Returns SUCCESS or FAILURE.

Examples

See also

Frequency, ftt, iftt

C++ prototype

```
Errc PFFTDeconvolution( const Img2duc &im_in1, const Img2duc
&im_in2, Img2dsf &im_in2 );
```

Version française

Déconvolution d’une image par un noyau.

Author: Régis Clouard

pfftshift

Shifts images in FFT image.

Synopsis

pfftshift [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out1*|-] [*im_out2*|-]

Description

pfftshift shifts the 4 subimages into the two input images. Thus, subimages 1,2 3,4 are shift to 4,3,2,1. The input images are considered as the real part and imaginary part of a complex image. The two input images must have the same size.

```
+--+--+      +--+--+
|1 |2 |      |4 |3 |
+--+--+  ->  +--+--+
|3 |4 |      |2 |1 |
+--+--+      +--+--+
```

Inputs

- *im_in1*: a 2D float image.
- *im_in2*: a 2D float image.

Outputs

- *im_out1*: a 2D float image.
- *im_out2*: a 2D float image.

Result

Returns SUCCESS or FAILURE.

Examples

Computes the magnitude of the Fast Fourier Transform of tangram.pan. The imaginary part (i1.pan) is null. (Use log transform dynamic in pvisu to display 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
```

See also

Frequency

C++ prototype

```
Errc FFTShift( const Img2dsf &im_in1, Img2dsf &im_in2, Img2dsf
&im_out1, Img2dsf &im_out2 );
```

Version française

Permutation des 4 sous-images de la transformée de Fourier.

Author: Régis Clouard

pfile

Displays properties of Pandore file.

Synopsis

pfile *im_in*

Description

pfile writes a short description about a Pandore file type.

For example, applied to an image file **pfile** writes the name of the creator, the creation date, the type of pixel, the number of row and columns, etc.

Inputs

- *im_in*: an image, a region map, a graph or a collection.

Result

No result values.

Examples

Displays information about the image file "tangram.pan".

```
pfile tangram.pan
```

See also

Information

Version française

Affichage des propriétés d’un fichier Pandore.

Author: Régis Clouard

pfillhole

Fills region holes.

Synopsis

pfillhole *connexity* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

pfillhole builds the output region map *rg_out* with the regions of the input region map *rg_in* where all holes are filled.

A hole is an inner region into a unique region with label =0 (ie, the inner region has only one neighbor). The *connexity* defines the conexity of the holes. If the continuity of holes is defined as 4-connexity then then continuity of the regions is defined as 8-connexity and conversely. A hole that touches the border are not considered as a hole. The hole is filled with the same label as the including region. Regions keep the same label in the output region map than in the input region map.

Parameters

- *connexity* specifies the hole connexity (4 and 8 for 2D image and 6 or 26 for 3D image).

Inputs

- *rg_in*: a region map.

Outputs

- *rg_out*: a region map.

Result

Returns the number of filled holes.

Examples

Fills holes (4-connex) of the regions in the input region map rin.pan:

```
pfillhole 4 rin.pan rout.pan
```

See also

Region

C++ prototype

```
Errc PFillHole( const Reg2d &rg_in, Reg2d &rg_out, int connexity );
```

Version française

Bouchage des trous dans les régions.

Author: Régis Clouard

pfisher

Performs multi-thresholding on image using Fisher algorithm.

Synopsis

pfisher *minval nbclass* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pfisher classifies the input image pixels into a small number of clusters according to their value. Every pixel p of the input image is assigned to a cluster identified by the related threshold value:

```
if threshold[k-1]<im_out[p]<=threshold[k].
then im_out[p]=threshold[k]
```

The last threshold is equal to 255.

The classification is based on the histogram partitioning into *nbclass* distinct classes so as to minimize the sum of the class variance.

Notice: This operator can only work on grayscale image of bytes.

Parameters

- *minval* is the minimum gray level from which the histogram is built. The value is generally 0 but can be used to mask some gray levels.
- *nbclass* specifies the number of class. It is an integer from [2 .. 25].

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).

Outputs

- *im_out*: a grayscale image of bytes (Img2duc, Img3duc).

Result

Returns SUCCESS or FAILURE.

Examples

Segments the tangram pieces:

```
pfisher 0 2 tangram.pan out.pan
```

See also

Thresholding

C++ prototype

```
Errc PFisher( const Img2duc &im_in, Img2duc &im_out, Uchar minval,
int nbclass );
```

Version française

Multiseuillage de l’image par partionnement de l’histogramme des niveaux de gris.

Author: Régis Clouard

pfits2pan

Converts FITS (Flexible Image Transport System) image file to Pandore image file.

Synopsis

pfits2pan [*im_in*|-] [*im_out*|-]

Description

pfits2pan reads data from the header and primary data of the FITS (Flexible Image Transport System) file *im_in* and then stores data in the output image *im_out*. Output image values are always stored as reals whatever is the original data format. Only 1D, 2D and 3D gray-scale images are considered at the moment.

Inputs

- *im_in*: a FITS image file.

Outputs

- *im_out*: a Pandore image file.

Result

Returns SUCCESS or FAILURE.

Examples

Converts the fits image to Pandore image:

```
pfits2pan image.fits image.pan
```

See also

Conversion, ppan2fits

C++ prototype

```
Pobject* PFits2Pan( const char *filename );
```

Version française

Conversion d’une image au format FITS vers le format Pandore.

Author: Jalal Fadili

pflip

Performs flip transformation for image or region map.

Synopsis

pflip *axis* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pflip builds a new image (or a region mp) *im_out* that is the symmetrical of the input image (or region map) *im_in* abut a specified *axis*.

For example, the flip transform of the input 2D image about the x axis in calculated as follows:

```
im_out[y][x]=im_in[y][width-x-1]
```

Parameters

- *axis* is an integer in the interval [0..2] where:
  - 0: flip about the x axis.
  - 1: flip about the y axis.
  - 2: flip about the z axis.

Inputs

- *im_in*: an image or a region map.

Outputs

- *im_in*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Builds the symmetrical of the tangram.pan image:

```
pflip 0 tangram.pan a.pan
```

See also

Transformation

C++ prototype

```
Errc PFlip( const Img3duc &im_in,Img3duc &im_out, int axis );
```

Version française

Construction du symétrique d’une image.

Author: François Angot

pfuzzyclustering

Performs pixel clustering onimage using fuzzy k-means algorithm.

Synopsis

pfuzzyclustering *nbclass fuzzy_degree* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

fuzzyclustering classifies the input image pixels into *nbclass* number of clusters. The method uses the fuzzy k-mean algorithm.

The result is region map *rg_out*.

Parameters

- *nbclass* is the number of cluster. Is is a positive value.
- *fuzzy_degree* specifies the fuzzy degree of the classification. It is a real value from [1..2] where 1 corresponds to no fuzzy.

Inputs

- *im_in*: an image.

Outputs

- *rg_out*: a region map.

Result

Returns SUCCESS or FAILURE.

Examples

Segments the tangram pieces:

```
pfuzzyclustering 2 1.5 tangram.pan out.pan
```

See also

Thresholding

C++ prototype

```
Errc PFuzzyClustering( const Img2duc &im_in, Reg2d &rg_out, int
nbclass, float fuzzy_degree );
```

Version française

Classification des pixels d’une image par la méthode des k moyennes floues.

Author: Jalal Fadili

pgaussaggregation

Performs pixel aggregation based on gaussian criterion.

Synopsis

pgaussaggregation *connexity alpha* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

pgausaggregation builds a new region map from aggregation of pixels to regions of the input region map *rg_in*. A pixel p is aggregated to a region R if:

- p is connected to the region R according to the specified *connexity*;
- [mean(R) - alpha*stdv(R), mean(R) + alpha*stdv(R)] ≤ threshold

where mean(R) is the inner mean of the region R and stdv(R) is the standard deviation of region R.

The mean and the standard deviation of the region are not updated with the new pixel to avoid moving away too much from the initial situation. One prefer iterative executions of the operator to update the mean and the standard deviation. For example, operator can be iterated until pstatus returns 0.

The output region map *rg_out* has the same number of labels than the input region map.

Parameters

- *connexity* specifies the neighbor relation between pixel and region (4 or 8 for 2D; 6 or 26 for 3D).
- *alpha* specifies the width of the gaussian. It is a real value. A typical value is 3.0.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of aggregation or FAILURE.

Examples

Aggregates pixels to tangram pieces:

```
pbinarization 96 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pgaussaggregation 8 4 b.pan tangram.pan out.pan
```

See also

Segmentation

C++ prototype

```
Errc PGaussAggregation(Reg2d &rg_in, Img2duc &im_in, Reg2d &rg_out,
int connexity, float alpha);
```

Version française

Croissance des régions d’une carte selon une distribution gaussienne.

Author: Régis Clouard

pgaussclassification

Performs gauss clustering on a set of objects.

Synopsis

pgaussclassification *attr_base attr_in attr_out* [*col_base*|-] [*col_in*|-] [*col_out*|-]

Description

pgaussclassification is a partitioning method for a group of n objects into *k* clusters. The basic idea supposes that the class distribution has a gaussian distribution, and for each object x to be classified the principle is to find the class that has the maximum probability to contain x.

Practically, `pgaussclassification` finds the class `i` that minimizes:

$$f(x,i) = \ln(\det A(i)) + {}^{t}(x - m(i)).A(i)^{-1}.(x - m(i)) - \ln(P(i)^{2})$$

- where `x` is the feature vector for the object x;
- `A(i)` is the covariance matrix for the class `i` ;
- `m(i)` is mean vector the for class `i` ;
- `P(i)` is the a priori probability to find class `i`.

These values can be calculated from the operator parraycovarmat.

Parameters

- *attr_base* is the base name for the gaussian features. If there exists n clusters and p features:
  - `attr_base.mean` is an array of `n*p` values which contains at the index [`i*n+j`] the mean of the `j+1`th feature of the `i-1` cluster.
  - `attr_base.det` is an array of n reals which contains at the index [`i-1`] the determinant `det(A(i))`.
  - `attr_base.inv` is an array of `p*p` values which contains at the index [`k*p*p + i*p +j`] the value of `k-1`th matrix cell of the A^{-1}[i,j]. (These 3 attributes can be calculted thanks to the operator parraycovarmat.) `attr_base.pap` is an array of n reals which contains a priori probabilities of each cluster. (This array can be omitted; in this case probabilities are supposed equiprobable).
- *attr_in* is the base name of the feature vector of the objects to be classified. The vectors are named `attr_in.1`, `attr_in.2`, ..., `attr_in.n` in the input collection. The item j of the array `attr_in.i` contains the (i)th feature of the (j+1)th object. They are Double arrays.
- *attr_out* is the name of the output array. Each item i of the array contains the cluster index from which the (i)th object is assigned. *attr_out* is an array of unsigned longs where attr_out[i] specifies the cluster index for the object i.

Inputs

- *col_base*: a collection which contains the learned parameters.
- *col_in*: a collection which contains the objects to be classified.

Outputs

- *col_out*: a collection which contains classified objects.

Result

Returns SUCCESS or FAILURE.

Examples

Classifies beans into the jellybean.pan image from sample of each bean stored in the directory ’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
```

See also

Classification

C++ prototype

```
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);
```

Version française

Classification utilisant un modèle gaussien.

Author: Alexandre Duret-Lutz

pgaussianfilter

Designs lowpass, highpass, bandpass or bandreject Gaussian filter.

Synopsis

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

Description

pgaussianfilter designs either a lowpass, highpass, bandpass or bandreject Gaussian filter. If *ndep<2* the filter *im_out* is a 2D float image with size nrow*ncol otherwise the filter *im_out* is a 3D float image with size ndep*nrow*ncol.

The Gaussian lowpass filter cuts off high-frequency components of the Fourier transform that are at a distance greater than a specified distance D0 (the *cutoff* value) from the origin of the centered transform.

The transfer function for a 2D Gaussian lowpass filter and with *cutoff* frequency at distance D0 from the origin is defined as:

$$Hlp(u,v) = \exp(-D^2(u,v)/2D0^2)$$

where D(u,v) is the distance of point (u,v) from the origin:

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

where N the number of rows and M the number of columns.

The transfert function for a Gaussian highpass is defined as:

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

The transfer function for a band reject is:

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

where Hhp(u,v) is the highpass filter with cutoff parameter and Hlp(u,v) is the lowpass filter with cutin parameter.

Parameters

- *ncol*, *nrow*, *ndep* specify the size of the output image. If *ndep*<1 then the output image *im_out* is a 2D image otherwise a 3D image.
- *highpass* is used in conjunction with the *cutin* parameter. It specifies the type fo the filter:
  - highpass=0 and cutin=0 : lowpass filter

    - highpass=1 and cutin=0 : highpass filter
    - highpass=0 and cutin=1 : bandreject filter
    - highpass=1 and cutin=1 : bandpass filter
- *cutin* is the cut in frequency of the filter D0 in case of bandreject or bandpass filter. In this case, the band width=cutoff-cutin and D0=(cutoff+cutin)/2.
- *cutoff* is the cutoff frequency of the filter D0. It must be a positive real value in the interval] 0.. sqrt(M*m+N*n)/2]. It corresponds to an euclidean distance from the center of the image. The higher cutoff is, the lower is the lowpass or the higher is the highpass.

Outputs

- *im_out*: a float 2D image (Img2dsf).

Result

Returns SUCCESS or FAILURE in case of bad parameter values.

Examples

Performs Gaussian lowpass filtering:

```
psetcst 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
```

Performs Gaussian highpass filtering:

```
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
```

Performs Gaussian bandreject filtering:

```
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
```

Performs Gaussian bandpass filtering:

```
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
```

See also

Frequency, pifft, pfftshift

C++ prototype

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

Version française

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

Author: Régis Clouard

pgaussianfiltering

Performs gaussian filtering on image.

Synopsis

pgaussianfiltering *sigma* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pgaussfiltering applies a gaussian filter to the input image *im_in*.

The input image is convoluted with the gaussian filter to produce the output image. The gaussian filter F(i) of size (6**sigma*) is built as follows:

```
F(i)= exp(-((Double)(i-halfsize)*(i-halfsize)/(2.0*sigma*sigma)))
```

where halfsize = sigma*3.

The image border (of size halfsize) is not considered for processing. The output image border is just a copy of the input image border.

Parameters

- *sigma* is the standard deviation of the gaussian. It is a real between [0 .. imagesize/6]. The greater is sigma, the stronger is the filtering. It is generally determined by the size of the objects inside the input image. Object smaller than 6*sigma are removed form the output image. A typical value is 1.0

Inputs

- *im_out*: an image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Applies a gaussian filter to tangram.pan:

```
pgaussianfiltering 1 tangram.pan out.pan
```

See also

Filtering

C++ prototype

```
Errc PGaussianFiltering( const Img2duc &im_in, Img2duc &im_out,
float sigma );
```

Version française

Lissage d’une image par une gaussienne.

Author: Régis Clouard

pgeodesicdilatation

Performs geodesic dilatation.

Synopsis

pgeodesicdilatation *num_se halfsize iteration* [*-m mask*] [*im_in*|-] [*im_msq*|-] [*im_out*|-]

Description

pgeodesicdilatation performs the dilation of the pixels of the image *im_in* as long as those pixels belong to a non null area specified in the image *im_msq*.

im_msq is an image of bytes or a region map used as a binary mask. All the non null pixels correspond to a true value for the mask.

The structuring element is specified by its type *num_se* and its size *halfsize*.

Geodesic dilation of point p corresponds to the operation:

```
if im_msq (p)!=0
   dilation(p) = MAX(neighbors of p specified by the structuring element)
else
   dilatation(p) = im_in(p).
```

Conditional dilation is defined as:

```
pdilatation hs in.pan i1.pan
pmask i1.pan msq.pan out.pan
```

For a binary image, dilatation dilates white areas.

For the images color, the lexicographic order is used: initially by using band X, in the event of equality by using the band Y then band Z.

Parameters

- *num_se* specifies the type of the structuring element:

  case of 2D:

  - 0: diamond (4-connexity)
  - 1: square (8-connexity)
  - 2: disc
  - 3: horizontal line (-)
  - 4: vertical line (|)

    - 5: right oblique line (/).
    - 6: left oblique line (\).

  case of 3D:

    - 0: bipyramid (6-connexity)
    - 1: cube (26-connexity)
    - 2: sphere
    - 3: line along X (-)
    - 4: line along Y (|)
    - 5: line along Z
- *halfsize* specifies the half-size of the structuring element. For example, a half-size of 1 for a square gives a structuring element of size 3x3.
- *iteration* specifies the number of iterations. If *iteration*=-1, then geodesic dilatation is carried out until idempotence.

Inputs

- *im_in*: a gray level or a color image.
- *im_msq*: an image of bytes or a region map.

Outputs

- *im_out*: an image of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Performs edge detection based on hysteresis thresholding:

```
pgradient 1 tangram.pan i1.pan i2.pan
pbinarization 80 1e30 i1.pan i3.pan
pbinarization 30 1e30 i1.pan i4.pan
pgeodesicdilatation 1 1 -1 i3.pan i4.pan out.pan
```

See also

Morphology, pgeodesicerosion

C++ prototype

```
Errc PGeodesicDilatation( const Img2duc &im_in, const Img2duc
&im_msq, Img2duc &im_out, int num_se, int halfsize, int iteration );
```

Version française

Dilatation géodésique des points de plus fort contraste de l’image.

Author: Régis Clouard

pgeodesicerosion

Performs geodesic erosion.

Synopsis

pgeodesicerosion *num_se halfsize iteration* [*-m mask*] [*im_in*|-] [*im_msq*|-] [*im_out*|-]

Description

pgeodesicerosion performs the erosion of the pixels of the image *im_in* as long as those pixels belong to a null area specified in the image *im_msq*. *im_msq* is an image of bytes or a region map used as a binary mask. All the non null pixels correspond to a true value for the mask. The structuring element is specified by its type *num_se* and its size *halfsize*.

Geodesic erosion of point p corresponds to the operation:

```
if im_msq (p)=0
   erosion(p) = MIN(neighbors of p specified by the structuring element)
else
   erosion(p) = im_in(p).
```

Whereas, conditional erosion is defined as :

```
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
```

For the color images, the lexicographic order is used: initially by using band X, in the event of equality by using the band Y then band Z.

Parameters

- *num_se* specifies the type of the structuring element:

  case of 2D:

  - 0: diamond (4-connexity)
  - 1: square (8-connexity)
  - 2: disc
  - 3: horizontal line (-)
  - 4: vertical line (|)
  - 5: right oblique line (/).
  - 6: left oblique line (\).

case of 3D:

  - 0: bipyramid (6-connexity)
  - 1: cube (26-connexity)
  - 2: sphere
  - 3: line along X (-)
  - 4: line along Y (|)
  - 5: line along Z
- *halfsize* specifies the half-size of the structuring element. For example, a half-size of 1 for a square gives a structuring element of size 3x3.
- *iteration* specifies the number of iterations. If *iteration*=-1, then geodesic dilatation is carried out until idempotence.

Inputs

- *im_in*: a gray level or a color image.
- *im_msq*: an image of bytes or a region map.

Outputs

- *im_out*: an image of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Performs edge detection based on hysteresis thresholding:

```
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
```

See also

Morphology, pgeodesicdilatation

C++ prototype

```
Errc PGeodesicErosion( const Img2duc &im_in, const Img2duc &im_msq,
Img2duc &im_out, int num_se, int halfsize, int iteration );
```

Version française

Erosion géodesique des points de plus fort contraste de l’image.

Author: Régis Clouard

pgetband

Gets a specified band in a multispectral image.

Synopsis

pgetband *band* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pgetband creates a grayscale image *im_out* with the specified band of the input image. The type of the grayscale image is the same as the input image.

Parameters

- *band* is an integer. If the value if lower or greater than the current number of bands in the input image than the closest band is used (the first or last band).

Inputs

- *im_in*: an image.

Outputs

- *im_out*: a grayscale image.

Examples

Extracts the first band of the parrot.pan color image:

```
pgetband 0 parrot.pan a.pn
```

Result

Returns SUCCESS or FAILURE.

See also

Utility

C++ prototype

```
Errc PGetBand( const Imc2duc &im_in, Img2duc &im_out, int band );
```

Version française

Récupération d’une bande donnée dans une image multispectrale.

Author: Régis Clouard

pgetslice

Gets slice from 3D image.

Synopsis

pgetslice *slice* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pgetslice builds the new 2D image *im_out* from the specified slice of the input 3D image *im_in*.

The result image *im_out* is of the same type as the input image.

Parameters

- *slice* specifies the index of the slice to be get. It is an integer between 0 and the total number of *im_in* slice minus 1. If *slice* is < 0 or > total number then the last slice is extracted.

Inputs

- *im_in*: a 3D image or 3D region map.

Outputs

- *im_out*: a 2D image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Gets slice no. 10 of the 3D image a3d.pan (the 11th slice):

```
pgetslice 10 a3d.pan a2d.pan
```

See also

Utility, psetslice, paddslice

C++ prototype

```
Errc PGetSlice( const Img3duc &im_in, Img2duc &im_out, long slice );
```

Version française

Construction d’une image 2D avec un plan d’une image 3D.

Author: Régis Clouard

pgetsubband

Gets one subband image from DWT image.

Synopsis

pgetsubband *scale subband* [*im_in*|-] [*im_out*|-]

Description

pgetsubband gets a subband from a DWT image at the specified *scale*. The output subband *im_out* is an image. At each scale, image are numbered as follows:

```
[1][2]
[3][4]
```

- 1: subband LL of approximate coefficients.
- 2: subband LH of detail coefficients.
- 3: subband HL of detail coefficients.
- 4: subband HH of detail coefficients.

Parameters

- *scale* specifies the scale analysis of the DWT image.
- *subband* specifies the number of the subband [1..4] to be extracted at the specified scale.

Inputs

- *im_in*: a 2D grayscale image.

Inputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Gets the LL and LH images of the DWT analysis of a square:

```
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
```

See also

Frequency

C++ prototype

```
Errc PGetSubband( const Img2dsf &im_in, Img2dsf &im_out, int scale,
int subband );
```

Version française

Extraction d’une sous-bande d’une image de DWT.

Author: Ludovic Soltys

pgif2pan

Converts GIF image file to Pandore image file.

Synopsis

pgif2pan *im_in* [*im_out*|-]

Description

pgif2pan converts GIF image file to Pandore image file. The result image can be a gray level 2D image of bytes (Img2duc) or a color 2D image of bytes (Imc2duc).

Note: Only uncompressed GIF image can be converted. It might be necessary to use an another image converter to convert compressed GIF file to uncompressed GIF file.

Inputs

- *im_in*: a GIF image file.

Outputs

- *im_out*: a Pandore image file.

Result

Returns SUCCESS or FAILURE.

Examples

Converts the GIF image to Pandore image:

```
pgif2pan image.gif image.pan
```

See also

Conversion, ppan2gif

C++ prototype

```
Errc PGif2Pan( const FILE* fdin, Pobject **objout );
```

Version française

Conversion d’une image GIF en image Pandore.

Author: Régis Clouard

pgr2im

Converts graph to float image.

Synopsis

pgr2im *type* [*gr_in*|-] [*im_out*|-]

Description

pgr2im builds the output image *im_out* from the input graph *gr_in*. Each node with a value > 0 is represented in the output image as a pixel with the related value at the related coordinates. Each edge is represented following the convention specified by the parameter *type*.

The output image *im_out* is a float grayscale image.

Parameters

- *type* specifies which edge to represent in the output image:
  - 0,1: edges with weight value > 0.
  - 2: edges that links two nodes with the same value.
  - 3- edges that links at least one node with value=0.

Inputs

- *gr_in*: a graph.

Outputs

- *im_out*: a float grayscale image.

Result

Returns SUCCESS or FAILURE.

Examples

Draws the graph a.pan in the image b.pan:

```
pgr2im 0 a.pan b.pan
```

See also

Casting

C++ prototype

```
Errc PGr2Im( const Graph2d &grs, Img2dsf &imd, int type );
```

Version française

Conversion d’un graphe en une image.

Author: Régis Clouard

pgr2rg

Converts graph to region map.

Synopsis

pgr2rg [*-m mask*] [*gr_in*|-] [*rg_in*|-] [*rg_out*|-]

Description

pgr2rg builds the output region map *rg_out* from the selection of regions of the input region map *rg_in* that match the nodes of the input graph *gr_in*. Only regions that are indexed by a node are kept in the output region map.

The label of the output region map are the same as the input region map. There is no relabeling.

Inputs

- *gr_in*: a graph.
- *rg_in*: a region map.

Outputs

- *rg_out*: a region map.

Result

Returns SUCCESS or FAILURE.

Examples

Builds the new region map rout.pan with the regions of rin.pan that are indexed by the node of the input graph g.pan:

```
prg2gr g.pan rin.pan rout.pan
```

See also

Casting

C++ prototype

```
Errc PGr2Rg( const Graph2d &gr_in, const Reg2d &rg_in, Reg2d &rg_out
);
```

Version française

Construction d’une carte de régions par application d’un graphe sur une carte de régions.

Author: Régis Clouard

pgradient

Computes the gradient magnitude and direction.

Synopsis

pgradient *halfsize* [*-m mask*] [*im_in*|-] [*im_out1*|-] [*im_out2*|-]

Description

pgradient computes the first derivative of the input image *im_in*. The result is two grayscale images, where each *im_out1*'s pixel is set to the magnitude value of the gradient at this point and each *im_out2*'s pixel is set to the direction of the gradient.

The gradient magnitude value reflects the amount of grayscale variation in this point. The more is the variation the greater is the value. The magnitude is the maximum between x derivative and y derivative (+ z derivative in 3D). The variation is computed inside a given neighborhood space size specified by the parameter *halfsize*.

The direction is the atan(dy/dx) follows by a discretization to get values in the freeman codes. So the output image is set with values from [0..7] in 2D and [0..25] in 3D.

Freeman codes are:

```
   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
```

The derivative is computed by convolution with the kernel in all directions and the magnitude is set with the maximum value:

```
-1, 0, 1
```

For color images, the Di Zenzo algorithm is used. It is based on the calculus of the eigen-values of the matrix:

```
  | 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
```

The gradient magnitude is given by:

```
magnitude=sqrt(lambda1 + lambda2)
with 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))
```

and direction is given by:

```
direction = 1/2 *arctan (2*t / (p-q))
```

followed by a discretization according to Freeman encoding.

Parameters

- *halfsize* specifies the half size of the convolution kernel.

Inputs

- *im_in*: an image.

Outputs

- *im_out1*: an image of the same type as im_in.
- *im_out2*: a Uchar image.

Result

Returns SUCCESS or FAILURE.

Examples

Performs an edge detection based on hysteresis threshold:

```
pgradient 1 tangram.pan i1.pan i2.pan
pbinarization 30 1e30 i1.pan i3.pan
pbinarization 60 1e30 i1.pan i4.pan
pgeodesicdilatation 1 1 -1 i4.pan i3.pan i4.pan
psuperimposition 1 tangram.pan i4.pan out.pan
```

See also

Edge detection

C++ prototype

```
Errc PGradient( const Img2duc &im_in, Img2duc &im_out1, Img2duc
&im_out2, int halfsize );
```

Version française

Calcul du module et de la direction du gradient par convolution.

Author: Régis Clouard

pgradientthreshold

Estimates the noise level in a gradient image.

Synopsis

pgradientthreshold *percent* [*-m mask*] [*im_in*|-]

Description

pgradientthreshold calculates the threshold value that is supposed to separate false contours to true contours. False contours are considered as gradient noise.

The threshold value is computed from the cumulated histogram of the positive gradient magnitude in *im_in*. The threshold value is the maximum gradient magnitude that represents 1-*percent* values of the cumulated histogram.

This value can be get by the way of the operator **pstatus**.

Parameters

- *percent* is an integer from [0..1] which specifies the amount of true contour points from the total number of points in the input image. A typical value is 0.2 which represents 20%.

Inputs

- *im_in*: an image.

Result

Returns the threshold value.

Examples

Extracts contours from the tangram.pan image:

```
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
pgeodesicdilatation 1 1 -1 i6.pan i7.pan out.pan
```

See also

Edge detection

C++ prototype

```
Errc PGradientThreshold( const Img2duc &im_in, float percent );
```

Version française

Estimation du bruit dans une image d’amplitude du gradient.

Author: Régis Clouard

pgradneumann

Computes the discrete gradient by forward finite differences and Neumann boundary conditions.

Synopsis

pgradneumann [*-m mask*] [*im_in*|-] [*im_out1*|-] [*im_out2*|-]

Description

pgradneumann computes the first derivative of the input image *im_in*. The result is two grayscale images, where *im_out1* is the derivative along the x axis, and *im_out2* is the derivative along the y axis:

```
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), with im_out1(n-1) = 0 and im_out2(n-1) = 0.
```

Inputs

- *im_in*: a 2D image.

Outputs

- *im_out1*: an image of the same type as im_in.
- *im_out2*: an image of the same type as im_in.

Result

Returns SUCCESS or FAILURE.

Examples

Implements the gradient and divergence operators with Neumann boundary conditions such that one is the adjoint of the other, i.e. <grad x,u> = <-div u,x>. The script checks this identity.

```
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
```

See also

Edge detection, pdivneumann

C++ prototype

```
Errc Errc PGradNeumann( const Img2d<U> &im_in, Img2d<U> &im_out1,
Img2d<U> &im_out2 );
```

Version française

Calcul du gradient d’une image par différences finies décentrées à droite avec conditions aux bords de Neumann.

Author: Jalal Fadili

pgraphneighbours

Valuates nodes with the number of neighbours.

Synopsis

pgraphneighbours [*gr_in*|-] [*gr_out*|-]

Description

pgraphneighbours sets every node value with the number of its neighbors. The structure of the output graph *gr_out* keeps the same structure than the input graph *gr_in*.

Inputs

- *gr_in*: a graph

Outputs

- *gr_out*: a graph.

Result

Returns SUCCESS or FAILURE.

Examples

```
pgraphneighbors g1.pan g2.pan
```

See also

Graph

C++ prototype

```
Errc PGraphNeighbours( const Graph &gr_in, Graph &gr_out );
```

Version française

Valuation des sommets d’un graphe avec le nombre de sommets voisins.

Author: François Angot

pgraphpruning

Performs graph pruning.

Synopsis

pgraphpruning [*-m mask*] [*gr_in*|-] [*gr_out*|-]

Description

pgraphpruning cuts edges between nodes that at least one node is null (value = 0).

Inputs

- *gr_in*: a graph.

Outputs

- *gr_out*: a graph.

Result

Returns SUCCESS or FAILURE.

Examples

```
pgraphpruning g1.pan g2.pan
```

See also

Graph

C++ prototype

```
Errc PGraphPruning( const Graph &gr_in, Graph &gr_out );
```

Version française

Suppression des arcs nuls et des sommets isolés.

Author: François Angot

pgraphvisu

Visualizes node and edge of graph.

Synopsis

pgraphvisu [*-m mask*] [*gr_in*|-] [*im_out*|-]

Description

pgraphvisu allows the visualization of the nodes and the edges of the input graph *gr_in* as an image. Nodes are drawn as dots and edges are drawn as lines between nodes.

The output image is a Float image.

Inputs

- *gr_in*: a graph.

Outputs

- *im_out* : a Float image.

Result

Returns SUCCESS or FAILURE.

Examples

```
pvisugraph g.pan out.pan
```

See also

Graph

C++ prototype

```
Errc PGraphVisu( const Graph2d &gr_in, Img2dsf &im_out );
```

Version française

Visualisation des valeurs des sommets et des arcs d’un graphe.

Author: Régis Clouard

pgray2bw

Converts gray scale image to black and white image.

Synopsis

pgray2bw [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pgray2bw is a mean to reduce the gray level range of images down to the 2 colors. It consists of mapping the original grey image into a binary image.

The technique consists in using a grey-level pattern (a dither matrix) image to be compared to the original image. If the pixel is greater than the pixel on the pattern, the output pixel is white (255), otherwise it is black (0).

The 16x16 dither matrix used in this operator is:

```
{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}};
```

For each pixel p of the input image:

```
if (im_in[p] >= matrix[p.y%16][p.x%16])
   imd[p]=255;
else
   imd[p]=0;
```

Inputs

- *im_in*: a 2D image of bytes (Img2duc).

Outputs

- *im_out*: a 2D image of bytes (Img2duc).

Result

Returns SUCCESS or FAILURE.

Examples

Transforms image ’tangram.pan’ to black and white image.

```
pgray2bw tangram.pan a.pan
```

See also

Color

C++ prototype

```
Errc PGray2BW( const Img2duc &im_in, Img2duc &im_out );
```

Version française

Conversion d’une image de niveaux de gris en une image binaire équivalente.

Author: Régis Clouard

pgray2falsecolor

Converts gray scale image to image with false colors./

Synopsis

pgray2falsecolor [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pgray2falsecolor is a mean to reduce to convert a gray scale gray level image to a color image by using false colors.

each pixel of the input image *im_in* is converted to 3 values: red, green,bleu iusing a specified lut.

For each pixel of the input image p:

```
imd.R[p]=lut[0][ims[p]
imd.V[p]=lut[1][ims[p]
imd.B[p]=lut[2][ims[p]
```

Parameters

- *lut* specified the look up table to be used.
  - 0: "rainbow"
  - 1: predefined lut with only 15 colors.

Inputs

- *im_in*: a gray level image.

Outputs

- *im_out*: a color image.

Result

Returns SUCCESS or FAILURE.

Examples

Transforms image ’tangram.pan’ with the rainbow lut.

```
pgray2falsecolor 0 tangram.pan a.pan
```

See also

Color

C++ prototype

```
Errc PGray2FALSECOLOR( const Img2duc &im_in, Img2duc &im_out, int
lutId );
```

Version française

Conversion d’une image de niveaux de gris en une image binaire équivalente.

Author: Régis Clouard

pgraylevel2depth

Converts gray levels to depth values.

Synopsis

pgraylevel2depth *depthmax* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pgraylevel2depth builds a 3D image from a 2D image, where the gray level are converted to depth. For example, the gray level 127 at coordinate (x,y) in the 2D image is copied along the 127 first slices at the same coordinate.

The parameter *depthmax* defines the depth of the output image *im_out*. All depths are normalized from that maximum depth.

The 3D image is built from the last slice. It means that clear objects occupy the first slices while dark objects occupy the last slices.

The algorithm is as follows:

```
for (y=0; y< normalize(im_in[x],depthmax); y++)
        im_out[y][x] = normalize(im_in[x],depth);
```

Parameters

- *depthmax* defines the maximum depth and also the depth of the output image *im_out*.

Inputs

- *im_in*: a 2D gray level image.

Outputs

- *im_out*: a 3D Uchar image (img3duc).

Result

Returns SUCCESS or FAILURE.

Examples

Builds the 3D image out.pan from the 2D image tangram.pan:

```
pgraylevel2depth 50 tangram.pan a.pan
```

See also

Utility, pdepth2graylevel

C++ prototype

```
Errc PGraylevel2Depth( const Img2duc &im_in, Img3duc &im_out, long depthmax );
```

Version française

Construction d’une image de reliefs 3D à partir d’une image 2D.

Author: Jean-Marie Janik

pharris

Performs Harris corner detection.

Synopsis

pharris *sigma kappa* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pharris is a corner detector. Corners are T, L or Y junctions or points with strong texture variation. Corners correspond to double discontinuities of the intensity function caused by discontinuities in the reflectance or the depth functions.

The principle of the algorithm is to compute the covariance matrix C(x,y):

```
    | S(Ix.Ix) S(Ix.Iy) |
C=|                     |
    | S(Ix.Iy) S(Iy.Iy) |
```

where Ix and Iy are image gradient components and S(x) is the sum symbol. The eigenvalues represent the major and minor axis of the elliptical approximation of the gradient vector distribution. If the smaller eigenvalue of the matrix is positive, it is considered as a corner.

To avoid to compute the eigenvalue, Harris proposes to compute the response function R(x,y) for each pixel:

```
R=Ixx*Iyy-Ixy*Ixy - kappa * (Ixx+Iyy) * (Ixx+Iyy)
```

and than to choose the interest point as local maxima of function R(x,y).

kappa is a tunable parameter which determines how ’edge-phobic’ the algorithm is. The value has to be determined empirically. In the literature values in the range 0.04 - 0.15 have been reported as feasible.

The size of research area for the local maxima is determined from the *sigma* parameter (size=6*sigma).

The result image is a float image that encodes for each pixel the strength of the response.

Parameters

- *sigma* is the standard variation of the gaussian. It is used to define the size of neighborhood where local maxima are extracted (size=6*sigma). A typical value is between [1..3].
- *kappa* determines how ’edge-phobic’ the algorithm is. The typical value estimated by Harris is 0.04.

Inputs

- *im_in*: a 2D image.

Outputs

- *im_dest*: a 2D Float image.

Result

Returns SUCCESS or FAILURE.

Examples

Extracts corners from image tangram.pan and superimposes corners on the initial image.

```
pharris 2 0.04 tangram.pan a.pan
pbinarization 1e4 1e30 a.pan b.pan
padd b.pan tangram.pan out.pan
```

See also

Points of interest

C++ prototype

```
Errc PHarris( const Img2duc &im_in, Img2dsf &im_out, float sigma,
float kappa );
```

Version française

Détection de points d’intérêt selon l’algorithme de Harris-Stephens.

Author: Régis Clouard

phermiterescale

Performs a rescaling of image using the Hermite algorithm.

Synopsis

phermiterescale *zoomx zoomy zoomyz* [*im_in*|-] [*im_out*|-]

Description

phermiterescale uses a convolution kernel to interpolate the pixel of the input image *im_in* in order to calculate the pixel value of the output image *im_out*. The interpolation consists in weigthing the input pixels influence on the output pixels. The weights are relative to the position of the output pixels and are given by the Hermite algorithm:

```
| (2*x - 3)*x*x + 1 if -1
```

phistogram

Creates histogram from grayscale image.

Synopsis

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

Description

phistogram computes the histogram of the input image *im_in*. An histogram is an array where the item i indicates the number of pixel with grayscale value i.

For signed image, histogram is shifted so as the item 0 indicates the number of min value, and the last item (max-min) the number of max value.

The output image *im_out* is long 1D image.

Inputs

- *im_in*: a grayscale image.

Outputs

- *im_out*: a 1D Long image.

Result

Returns SUCCESS or FAILURE.

Examples

Builds the histogram of tangram.pan:

```
phistogram tangram.pan a.pan
pplot1d 256 256 0 0 0 a.pan b.pan
pvisu b.pan
```

See also

Image Features Extraction

C++ prototype

```
Errc PHistogram( const Img2duc &im_in1, Img1dus &im_in2 );
```

Version française

Création de l’histogramme d’une image.

Author: Alexandre Duret-Lutz

phistogramequalization

Performs contrast stretching using histogram equalization.

Synopsis

phistogramequalization *min max* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

phistogramequalization performs a gray-levels transformation by forcing the gray-levels to spread over the entire intensity range between *min* and *max*. As a consequence, histogram equalization expands gray-levels where information is important and compresses gray-levels where the information is low.
This transform is not likely to work well on input images that have dark gray-levels distribution.

The new output bounds of the output gray-levels are given by the parameter *min* and *max*.

The equalization has the form:

1. Compute the cumulated histogram for *im_in*;
2. Normalize histogram between [0..1];
3. Build output image *im_out* with for each pixel 'p':

```
im_out[p] = HC[im_in[p]]*(max-min);
```

For color and multispectral images, the transform uses the marginal approach: it is applied on each band individually.

Parameters

- *min* and *max* specify the bounds of the output pixel value. They are related to the type of the input image.
  Note: if *min* > *max* then min and max are set with the minimum and maximum values of the input image type; for example, 0 and 255 for Uchar images.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image with the same properties as im_in.

Result

Returns SUCCESS or FAILURE in case of invalid parameter values.

Examples

Equalizes histogram with the new bounds [0,255]:

```
phistogramequalization 20 200 tangram.pan a.pan
```

Equalizes histogram with the same bounds than the input image:

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

See also

Lut transform, phistogramspecification

C++ prototype

```
Errc PHistogramEqualization( const Img2duc &im_in, Img2duc &im_out,
float min, float max );
```

Version française

Rehaussement de contraste par égalisation d’histogramme.

Author: Régis Clouard

phistogramspecification

Performs contrast stretching using histogram specification.

Synopsis

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

Description

phistogramspecification performs a gray-levels transformation of the input image *im_in1* by using a particular histogram shape given by the reference image *im_in2*.

This operator is useful to enhance a list of images of the same scene. The first step is to enhance histogram of one of the image using particular techniques, and the other step is to apply the same histogram to the remainder images of the list.

The histogram specification is based on the following algorithm:

1. compute normalized cumulated histogram hc1 of input image *im_in1*;
2. compute normalized cumulated histogram hc2 of the reference image *im_in2*;
3. for each pixel ’p’ of the input image:
    1. s=hc1[im_in1[p]];
    2. Search for i such as hc2[i]=s;
    3. im_out[p]=i.

For color and multispectral images, the transform uses the marginal approach: it is applied on each band individually.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image with the same properties as im_in.

Result

Returns SUCCESS or FAILURE in case of incompatible input images.

Examples

Applies the histogram transform to tangram.pan from a reference image built with a logarithmic transform of tangram.pan.

```
plogtransform 0 0 255 tangram.pan reference.pan
phistogramspecification tangram.pan reference.pan a.pan
```

See also

Lut transform, phistogramequalization

C++ prototype

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

Version française

Rehaussement de contraste par spécification d’histogramme.

Author: Régis Clouard

phistomerging

Perfoms priority region merging based on histogram correlation.

Synopsis

phistomerging *number threshold* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

phistomerging merges connected regions of the input region map *rg_in* if the correlation between their histogram is greater than the specified *threshold*.

The algorithm uses the priority merging that consists in merging regions with the greater correlation.

The output region map *reg_out* defines the new regions and the output graph *gr_out* defines the new relationship between regions.

The correlation between two histograms is given by:

```
correlation(H1,H2) = H1.H2 / (norm(H1).norm(H2))
```

where H1.H2 = scalar product between H1 and H2 and norm(Hi) = euclidean norm of histogram Hi.

The greater is the correlation, the closer is the two histograms.

Parameters

- *number* specifies the number of allowed merging. If *number* = -1 then all possible merging are done.
- *threshold* specifies the maximum difference allowed between two regions to decide to merge them. Values belongs to [0..1], where 1 corresponds to two equal histograms. Typical values are close to 1 (e.g., 0.7).

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.
- *gr_out*: a graph.

Result

Returns the number of merging.

Examples

Merges regions yielded by a quadtree splitting process:

```
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
```

See also

Segmentation

C++ prototype

```
Errc PHistoMerging(Reg2d &rg_in, Graph2d &gr_in, Img2duc &im_in,
Reg2d &rg_out, Graph2d &gr_out, long number, float threshold);
```

Version française

Fusion prioritaire de régions selon la corrélation d’histogramme.

Author: Laurent Quesnel

phistothresholding

Performs multi-thresholding on image using histogram analysis.

Synopsis

phistothresholding *length* [*im_in*|-] [*im_out*|-]

Description

phistothresholding classifies the input image pixels into a small number of clusters according to their value. Every pixel p of the input image is assigned to a cluster identified by the related threshold value:

```
if threshold[k-1]<im_out[p]<=threshold[k].
then im_out[p]=threshold[k]
```

The last threshold is equal to 255.

The number of clusters and the value of the thresholds are determined from the histogram. Thresholds are located as regional maxima of the histogram. The maxima are searched in the space of *length* gray levels around the gray level i.

Notice: This operator can only work on grayscale image of bytes.

Parameters

- *length* defined the length of the search space of the regional minima. It is defined in gray level unit. The greater is the length, the less there are thresholds. A typical value is 10.

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).

Outputs

- *im_out*: a grayscale image of bytes (Img2duc, Img3duc).

Result

Returns the number of thresholds.

Examples

Segments the tangram pieces:

```
phistothresholding 10 tangram.pan out.pan
```

See also

Thresholding

C++ prototype

```
Errc PHistoThresholding( const Img2duc &im_in, Img2dsl &im_out, int length );
```

Version française

Multi-seuillage d’une image de niveaux de gris par une ligne de partage des eaux de son histogramme.

Author: Olivier Lezoray

phomotopicskeletonization

Computes the homotopic skeleton of 3D objects.

Synopsis

phomotopicskeletonization *connexity* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

phomotopicskeletonization computes the skeleton of the binary objects in the 3D image *im_in*. Binary objects are regions with connected non null pixels.

The algorithm rests on a sequential suppression of the simple points. A simple points is a point whose suppression preserves the topology of the image.

Parameters

- *connexity* specifies the number of neighbors for a pixel: 6 or 26.

Inputs

- *im_in*: a 3D Uchar grayscale image.

Inputs

- *im_out*: a 3D Uchar grayscale image.

Result

Returns SUCCESS or FAILURE.

Examples

Builds the homotopic skeleton of a parallelepiped (it will be reduced to a point):

```
pshapedesign 256 256 128 13 120 80 a.pan
phomotopicskeletonization 26 a.pan out.pan
```

See also

Morphology, pskeletonization

C++ prototype

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

Version française

Squelettisation homotopique d’objets binaires.

Author: Sébastien Fourey

phoughlines

Detects straight lines from a set of contours.

Synopsis

phoughlines *lines minangle maxangle* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

phoughlines detects straight lines from contours given in the input image *im_in*. A contour is a chain of connected non null pixels. The output image is composed of detected straight lines.

Hough Line Transform:

The Hough transform is a general technique for identifying the location and orientation of certain types of features in a digital image.

To use the Hough transform to extract line, we consider the general equation of straight line in normal form:

```
x cos theta + y sin theta = rho.
```

where rho is the perpendicular distance from the origin and theta the angle with the normal.

For any given point (x, y), we can obtain lines passing through that point by solving for rho and theta. A line in the image is represented as a point in the polar coordinates (rho, theta). Conversely, a point in the image is represented as a sinusoid in the polar coordinates since infinitely many lines pass through this point.

The hough transform is based on an accumulator (rho, theta). Each cell of the accumulator is the number of occurrence (rho, theta) for points of the perpendicular line, ie. the number of lines with the same parameters (rho, theta) that can passed through each contour of the input image.

The algorithm is as follows:

- initialize the accumulator
- For each point (x,y) in the image, increment accumulator(r,o) for each possible line that passes through (x,y):

```
for theta =0 to 360 do
   rho =x*cos(theta)+y*sin(theta)
   accumulator[rho][theta] ++;
```

- For the number of wanted *lines*:
  - Find maximum in the accumulator.
  - Draws related line in the output image.

- Remove the maximum in the accumulator

To solve the problem of the "phantom lines" due to the discretization of the contour lines, the maximum is not only removed, but all the contour points of the detected line are removed from the initial image and the accumulator is recomputed with the remain lines (in fact, an improvement of this solution).

Parameters

- *lines* specifies the number of lines in the output image.
- *minangle* and *minangle* control the angle of research. Only lines that are comprise between these two angle are retained. The values are specified in degree unit and are between -360 + 360. Uses minangle=0 and maxangle=180 to retain all lines.

Inputs

- *im_in*: a 2D grayscale image (Img2duc).

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns the number of detected lines.

Examples

Extracts straight lines from the set of contours yielded by a simple edge detection of tangram.pan. The result is superimposed to the image of contours :

```
psobel tangram.pan b.pan
pbinarization 45 1e30  b.pan c.pan
pskeletonization c.pan d.pan
ppostthinning d.pan e.pan
phoughlines 10 0 180 e.pan f.pan
pimg2imc 0 f.pan c.pan c.pan out.pan
```

See also

Contour

C++ prototype

```
Errc PHoughLines( const Img2duc &im_in, Img2duc &im_out, int lines,
int minagle, int maxangle );
```

Version française

Détection et localisation des segments de droite dans une image de contours par la transformée de Hough.

Author: Laurent Quesnel

phsi2rgb

Converts HSI color image to RGB color image.

Synopsis

phsi2rgb [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

phsi2rgb converts color image from the color space HSI (Hue, Saturation, Intensity) to the color space RGB (Red, Green, Blue).

Inputs

- *im_in*: a HSI color image.

Outputs

- *im_out*: a RGB color image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts parrot.pan from rgb to hsi and conversely:

```
prgb2hsi parrot.pan a.pan
phsi2rgb a.pan b.pan
```

See also

Color, prgb2hsi

C++ prototype

```
Errc PHSI2RGB( const Imc2dsf &im_in, Imc2dsf &im_out );
```

Version française

Changement d’espace couleur de HSI vers RGB.

Author: Olivier Lezoray

phsl2rgb

Converts HSL color image to RGB color image.

Synopsis

phsl2rgb [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

phsl2rgb converts color image from the color space HSL (Hue, Saturation, Luminosity) to the color space RGB (Red, Green, Blue).

A hue refers to the gradation of color within the visible spectrum, or optical spectrum, of light. It is expressed in degree unit [0..360].

Saturation or purity is the intensity of a specific hue: a highly saturated hue has a vivid, intense color, while a less saturated hue appears more muted and gray. With no saturation at all, the hue becomes a shade of gray. It is expressed as percentage [0..100].

Lightness is the amount of light in a color. It is expressed in gray level unit [0..255].

The conversion uses the following transformation:

```
q= | l * (1+s),  if l <  1/2
   | l+s - (l*s) if 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

for each C in {R,V,B}
  if t_C < 0 : t_C = t_C + 1.0
  if t_C > 1 : t_C = t_C - 1.0

      | p+((q-p)*6*t_C)        if t_C <1/6
      | q                     if 1/6 <= T_C <1/2
  C = | p+ ((q-p)*6*(2/3-T_C)) if 1/2 <= T_C < 2/3
      | p                     else
```

Inputs

- *im_in*: a HSL color image.

Outputs

- *im_out*: a RGB color image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts parrot.pan from rgb to hsl and conversely:

```
prgb2hsl parrot.pan a.pan
phsl2rgb a.pan b.pan
```

See also

Color, prgb2hsl

C++ prototype

```
Errc PHSL2RGB( const Imc2dsf &im_in, Imc2dsf &im_out );
```

Version française

Changement d’espace couleur de HSL vers RGB.

Author: RÃ©gis Clouard

pidwt

Performs inverse Direct Wavelet Transform.

Synopsis

pidwt *scale* [*im_in*|-] [*col_in*|-] [*im_out*|-]

Description

pidwt performs inverse wavelet transform on the input image *im_in* from the quadratic mirror filter stored in the input collection *col_in*.

The wavelet transform can be built from operator pdwt. And the filter can be generated from the operator pqmf.

Parameters

- *scale* specifies the number of levels used for the decomposition of the input image.

Inputs

- *im_in*: a 2D float image.
- *col_in*: a collection that contains the filter coefficients.

Ouputs

- *im_out*: a 2D float image.

Result

Returns SUCCESS or FAILURE.

Examples

Builds a synthetic image (a square) to illustrate the Gibbs phenomenon in wavelets analysis.

```
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
```

See also

Frequency, dwt, pqmf

C++ prototype

```
Errc PIdwt( const Img2dsf &ims, const Collection &c, Img2dsf &imd,
int scale=1 );
```

Version française

Reconstruction d’une image décomposée en ondelettes dyadiques biorthogonales.

Author: Ludovic Soltys

pifft

Performs inverse Fast Fourier Transform.

Synopsis

pifft [-m mask] [*im_in1*|-] [*im_in2*|-] [*im_out1*|-] [*im_out2*|-]

Description

ipfft computes the inverse Fast Fourier Transform of the input complex image *im_in1*. The input complex image is composed of two images:

- *im_out1* is the real part of the transform;
- *im_out2* is the imaginary part of the transform.

The output complex image is composed of two images:

- *im_in1* is the real part;
- *im_in2* is the imaginary part.

Fast Fourier Transform is a way of going from the frequency domain to the spatial domain.

Inputs

- *im_in1*: a gray level image (the real part of the transform).
- *im_in2*: a gray level image with the same properties than *im_in1* (the imaginary part of the transform).

Outputs

- *im_out1*: a gray level image (the real part of the transform).
- *im_out2*: a gray level image (the imaginary part of the transform).

Result

Returns SUCCESS or FAILURE.

Examples

Converts an image of a square from the spatial domain to the frequency domain and reciprocally:

```
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
```

See also

Frequency, pfft

C++ prototype

```
Errc IFFT( const Img2duc &im_in1, const Img2duc &im_in2, Img2dsf
&im_out1, Img2dsf &im_out2 );
```

Version française

Transformée de Fourier Rapide Inverse d’une image.

Authors: Herissay & Berthet

pim2array

Converts image to pixel array.

Synopsis

pim2array *name* [*im_in*|-] [*col_out*|-]

Description

pim2array builds the collection *col_out* with the array named *name* that contains the pixels of the input image *im_in*.

For the color image, the collection contains three arrays named *name.1*, *name.2* et *name.3* that contains respectively the pixels of the three bands red, green and blue.

Parameters

- *name* is the name of the array in the collection.

Inputs

- *im_in*: an image.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Converts the image tangram.pan to the vector foo in the collection col.pan:

```
pim2array foo tangram.pan col.pan
```

See also

Casting, parray2im

C++ prototype

```
Errc PIm2Array( const Img2duc &im_in, Collection &cd,std::string
name );
```

Version française

Conversion d’une image en un vecteur dans une collection.

Author: Alexandre Duret-Lutz

pim2d23d

Converts 2D image or region map to 3D image or region map with 1 slice.

Synopsis

pim2d23d [*im_in*|-] [*im_out*|-]

Description

pim2d23d builds a 3D image (or region map) from a 2D image (or region map). The output 3D image contains only one slice.

If the input is a 2D region map the output is a 3D region map with one slice.

The operator psetslice can be used to insert a 2D image into a 3D image.

Inputs

- *im_in*: a 2D image or a region map.

Outputs

- *im_out*: a 3D image or a 3D region map.

Result

Returns SUCCESS or FAILURE.

Examples

Converts tangram.pan to 3D image:

```
pim2d23d tangram.pan a.pan
```

See also

Casting, pim3d22d

C++ prototype

```
Errc PIm2d23d( const Img2duc &im_in, Img3duc &im_out );
```

Version française

Construction d’une image 3D à partir d’une image 2D.

Author: Régis Clouard

pim2rg

Converts grayscale image to region map.

Synopsis

pim2rg [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

pim2rg builds the region map *rg_out* from the grayscale image *im_in* . Each pixel value is converted to a label of the region map. Negative pixels are not considered.

Warning: There is no relabeling of regions. The label is set to the value of the related the pixel value.

Inputs

- *im_in*: an image.

Outputs

- *rg_out*: a region map.

Result

Returns the maximum label value in the result region map or FAILURE

Examples

Builds a region map from binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
pim2rg a.pan b.pan
```

See also

Casting

C++ prototype

```
Errc PIm2Rg( const Img2duc &im_in, Reg2d &rg_in );
```

Version française

Création d’une carte de régions à partir d’une image d’étiquettes.

Author: Régis Clouard

pim2sf

Converts any image type to float image.

Synopsis

pim2sf [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pim2sl builds a new unsigned char image from an image of any type. Each pixel of the input image is converted to unsigned char by using only a casting operation. There is no interpolation of the pixel values:

```
pixel(im_out)=(float)pixel(im_in);
```

Inputs

- *im_in*: an image.

Outputs

- *im_out*: a float image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts tangram.pan to float image:

```
pim2sf a.pan out.pan
```

See also

Casting

C++ prototype

```
Errc PIm2Sf( const Img2d &im_in, Img2duc &im_out );
```

Version française

Conversion automatique d’une image de n’importe quel type en type float.

Author: Régis Clouard

pim2sl

Converts any image type to signed long image.

Synopsis

pim2sl [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pim2sl builds a new signed long image from an image of any type. Each pixel of the input image is converted to signed long value by using only a casting operation. There is no interpolation of the pixel values:

```
pixel(im_out)=(long)pixel(im_in);
```

Inputs

- *im_in*: an image.

Outputs

- *im_out*: a signed long image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts tangram.pan to signed long image:

```
pim2sl tangram.pan out.pan
```

See also

Casting

C++ prototype

```
Errc PIm2Sl( const Img2d &im_in, Img2dul &im_out );
```

Version française

Conversion automatique d’une image de n’importe quel type en type signed long.

Author: Régis Clouard

pim2uc

Converts any image type to unsigned char image.

Synopsis

pim2uc [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pim2sl builds a new unsigned char image from an image of any type. Each pixel of the input image is converted to unsigned char by using only a casting operation. There is no interpolation of the pixel values:

```
pixel(im_out)=(unsigned char)pixel(im_in);
```

Inputs

- *im_in*: an image.

Outputs

- *im_out*: a unsigned char image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts the float image a.pan to Uchar image b.pan:

```
pim2uc a.pan b.pan
```

See also

Casting

C++ prototype

```
Errc PIm2Uc( comst Img2d &im_in, Img2duc &im_out );
```

Version française

Conversion automatique d’une image de n’importe quel type en type unsigned char.

Author: Régis Clouard

pim3d22d

Converts 3D image with one slice to 2D image.

Synopsis

pim3d22d [*im_in*|-] [*im_out*|-]

Description

pim3d22d builds a 2D image from a 3D image that only contains one slice. If the input is a 3D region map with one slice the output image is 2D region map.

The operator pgetslice can be used to extract a 2D slice from a 3D image.

Inputs

- *im_in*: a 3D image or a 3D region map.

Outputs

- *im_out*: a 2D image or a 2D region map.

Result

Returns SUCCESS or FAILURE.

Examples

Converts 3D image a.pan with 1 slice to 2D image b.pan:

```
pim3d22d a.pan b.pan
```

See also

Casting, pim2d23d

C++ prototype

```
Errc PIm3d22d( const Img3duc &im_in, Img2duc &im_out );
```

Version française

Construction d’une image 2D à partir d’une image 3D à un seul plan.

Author: Régis Clouard

pimc2img

Converts color image to grayscale image.

Synopsis

pimc2img *numband* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pimc2img creates a grayscale image *im_out* from one band of the input color image *im_in*. The type of the grayscale image is the same as the color image.

Parameters

- *numband* is an integer from [0..2]:
  - 0: the first band (for example the red band for RGB image, or the hue band for HSL image).
  - 1: the second band (for example the green band for RGB image, or the saturation band for HSL image).
  - 2: the third band (for example the blue band for RGB image, or the lightness band for HSL image).

Inputs

- *im_in*: a color image.

Outputs

- *im_out*: a grayscale image.

Examples

Extracts the first band of the parrot.pan color image:

```
pimc2img 0 parrot.pan a.pn
```

Result

Returns SUCCESS or FAILURE.

See also

Casting

C++ prototype

```
Errc PImc2Img( const Imc2duc &im_in, Img2duc &im_out, int noplan );
```

Version française

Construction d’une image de niveaux de gris avec un plan d’une image couleur.

Author: Régis Clouard

pimc2imx

Creates a multispectral image with a color image.

Synopsis

pimc2imx [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pimc2imx creates a new image multispectral image with 3 bands from a color image.

The type of the multispectral image *im_out* is the same as the color image *im_in*.

Inputs

- *im_in*: a color image.

Outputs

- *im_out*: a multispectral image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts the color image parrot.pan to the multispectral image a.pan:

```
imc2imx parrot.pan a.pan
```

See also

Casting

C++ prototype

```
Errc PImc2Imx( const Imc2duc &im_in, Imx2duc &im_out );
```

Version française

Construction d’une image multipectrale à 3 bandes à partir d’une image couleur.

Author: Régis Clouard

pimg2imc

Creates a color image from 3 grayscale images.

Synopsis

pimg2imc *colorspace* [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_in3*|-] [*im_out*|-]

Description

pimg2imc builds a color image from 3 grayscale images. Each input image *im_in1*, *im_in2*, *im_in3* corresponds to one band of the color image. *im_in1* becomes the first band, *im_in2* the second band and *im_in3* the third band in the color image. The three input images must have the same type and the same dimensions.

Parameters

- *colorspace* is an integer that specifies the colorspace:
  - 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

Inputs

- *im_in1*: a grayscale image.
- *im_in2*: a grayscale image.
- *im_in3*: a grayscale image.

Outputs

- *im_out*: a color image.

Result

Returns SUCCESS or FAILURE.

Examples

Builds the RGB color image out.pan where a.pan is the red band, b.pan is the green band and c.pan is the blue band.

```
pimg2imc 0 a.pan b.pan c.pan out.pan
```

See also

Casting

C++ prototype

```
Errc PImg2Imc( const Img2duc &im1, const Img2duc &im2, const Img2duc &im3, Imc2duc &imd, int colorspace);
```

Version française

Construction d’une image couleur à partir de trois images de niveaux de gris.

Author: Régis Clouard

pimg2imx

Creates a multispectral image from several grayscale images.

Synopsis

pimg2imx *dimension* [*-m mask*] [*im_in*|-]* [*im_out*|-]

Description

pimg2imx creates a multispectral image from the given list of grayscale images. The parameter *dimension* specifies the number of images and also provides the number of bands in the output image *im_out*.

Each image corresponds to a band in the multispectral keeping the order given in the argument list of the operator.

All input images must have the same type and the same dimension. The type of the multispectral image *im_in* is the same than all inputs images.

Parameter

- *dimension* specifies the number of input images. It is a positive integer.

Inputs

- *im_in**: grayscale images.

Outputs

- *im_out*: a multispectral image.

Result

SUCCESS or FAILURE if image are incompatible.

Examples

Creates a multispectral image with two bands:

```
pimg2imx 2 tangram.pan lena.pan a.pan
```

See also

Casting

C++ prototype

```
Errc Img2Imx( const Pobjects *im_in[], Imx2duc &im_out );
```

Version française

Creation d’une image multispectrale à partir d’images en niveaux de gris.

Author: Régis Clouard

pimx2imc

Converts multispectral image to color image.

Synopsis

pimx2imc *colorspace* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pimx2imc converts the multispectral image *im_in* to the color image *im_out*. Only multispectral image with less or equal to 3 bands can be converted. If input image have less than 3 bands, others bands are set to 0.

The color space of the result image is specified from the parameter *colorspace*.

Parameters

- *colorspace* is an integer that specifies the colorspace:
  - 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

Inputs

- *im_in*: a multispectral image.

Outputs

- *im_out*: a color image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts multispectral image a.pan to RGB color image b.pan:

```
pimx2imc 0 a.pan b.pan
```

See also

Casting

C++ prototype

```
Errc PImx2imc( const Imx2duc &im_in, Imc2duc &im_out, int colorspace
);
```

Version française

Construction d’une image couleur à partir d’une image multispectrale à 3 bandes.

Author: Régis Clouard

pimx2img

Converts one band of multispectral image to grayscale image.

Synopsis

pimx2img *numband* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pimx2img extracts the band *numband* from the multispectral *im_in* to create the grayscale image *im_out*.

Parameters

- *numband* is an integer that specifies the band number to be converted. If *numband* is greater than the total number of bands than the last band is selected.

Inputs

- *im_in*: a multispectral image.

Outputs

- *im_out*: a color image.

Examples

Extracts band #0 from a.pan:

```
pimx2img 0 a.pan b.pan
```

Result

Returns SUCCESS or FAILURE.

See also

Casting

C++ prototype

```
Errc PImx2Img( const Imx2duc &im_in, Img2duc &im_out, int numband );
```

Version française

Construction d’une image couleur à partir d’une image multispectrale à 3 bandes.

Author: Régis Clouard

pinnermerging

Performs inner region merging.

Synopsis

pinnermerging [*-m mask*] [*rg_in1*|-] [*rg_in2*|-][*rg_out*|-]

Description

pinnermerging merges regions of input region map *rg_in1* to regions of input region map *rg_in2* if *rg_in2* regions are inner *rg_in1* regions.

An inner region in *rg_in2* has only one neighbor region in region map *rg_in1* other than region 0. Regions of the region map *rg_in2* that have no neighbor in the region map *rg_in1* are not kept in the output region map *rg_out*.

Inputs

- *rg1_in*: a region map.
- *rg2_in*: a region map.
- *gr_in*: a graph.

Outputs

- *rg_out*: a region map.
- *gr_out*: a graph.

Result

Returns the number of merging.

Examples

Segments tangram.pan and then merges inner regions:

```
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
```

See also

Segmentation

C++ prototype

```
Errc PInnerMerging( const Reg2d &rg_in1, const Reg2d &rg_in2, Reg2d
&rg_out );
```

Version française

Fusion de régions englobées dans d’autres régions.

Author: Régis Clouard

pinnermostmerging

Performs inner-most region merging.

Synopsis

pmostinnermostmerging [*-m mask*] [*rg_in1*|-] [*rg_in2*|-] [*rg_out*|-]

Description

pinnermostmerging merges regions of input region map *rg_in1* to regions of input region map *rg_in2* if *rg_in2* regions are inner-most *rg_in1* regions.

Inner region in *rg_in2* is merged with the including region of input region map *rg_in* that has the longest common boundary points with the inner region other than region 0. Regions of region map *rg_in2* that have no neighbor in the region map *rg_in1* are not kept in the output region map *rg_out*.

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.

Outputs

- *rg_out*: a region map.
- *gr_out*: a graph.

Result

Returns the number of merging.

Examples

Segments tangram.pan and then merges inner regions:

```
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
```

See also

Segmentation

C++ prototype

```
Errc PMostInneMostMerging( const Reg2d &rg_in1, const Reg2d &rg_in2,
Reg2d &rg_out );
```

Version française

Fusion de régions avec la région voisine la plus englobante.

Author: Régis Clouard

pinnerselection

Selects inner regions.

Synopsis

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

Description

pinnerselection selects regions that are entirely included in an other region.

A region that touches the border or that touches the background (ie., the region with label=0) is not considered as inner region.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *rg_out*: a 2D region map.

Result

Returns the number of selected regions.

Examples

Selects inner regions from rin.pan:

```
pinnerselection rin.pan rout.pan
```

See also

Region

C++ prototype

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

Version française

Sélection des régions englobées dans une autre région.

Author: Régis Clouard

pinsertsubimage

Inserts subimage into image.

Synopsis

pinsertsubimage *x y z* [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

pinsertsubimage builds a new image *im_out* by inserting image *im_in2* into image *im_in1* at coordinates (*x*,*y*, *z*). Insertion is done by replacing pixels of *im_in1* with pixels of *im_in2*.

For region map, regions of *im_in2* are added to image *im_in1* with label values greater than the maximum label value of *im_in1*. Thus, the maximum label value of output image *im_out* is the maximum label of *im_in1* + the maximum label of *im_in2*.

Parameters

- *x, y, z* specify the coordinates in image *im_in1* of the higher left corner of image *im_in2*. In case of 2D image, *z* parameter is ignored but must be given.

Inputs

- *im_in*: an image or a region map.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

For image returns SUCCESS or FAILURE. For region map returns the maximum label of FAILURE.

Examples

Inserts image a.pan into tangram.pan:

```
pinsertsubimage 10 10 0 a.pan tangram.pan b.pan
```

See also

Utility, pextractsubimage

C++ prototype

```
Errc PInsertSubImage( const Img3duc &im_in1, const Img3duc &im_in2,
Img3duc &im_out, Long cz, Long cy, Long cx );
```

Version française

Insertion d’une image dans une autre image.

Author: Régis Clouard

pinterregioncontrast

Computes the goodness measure based on inter-region contrast.

Synopsis

`pinterregioncontrast` [*-m mask*] [*rg_in*|-] [*im_in*|-]

Description

pinterregioncontrast computes the inter-region contrast criterion as defined by M. Levine & A. Nazif[*]. This criterion relies on the idea that a good segmentation has a strong contrast between adjacent regions.

This criterion is near 1 when regions are homogeneous and near 0 when regions are heterogeneous.

The inter-region contrast is computed as follows:

$$criterion = \sum_R [\ AiCi / \sum Ai\ ] \qquad Ci = sum\ [\ (lij*|mi-mj|)\ /\ (li*(mi+mj))\ ]$$

where

- Ai is the area of the region i
- Ci is the contrast of the region i computed as follows:
- mi is the mean of region i.
- mj is the mean of the region j adjacent to i,
- li is the perimeter of the region i
- lij the length of the boundary between region i and region j.

Caution: Regions with label=0 are not considered for computing.

Inputs

- *rg_in*: a region map.
- *im_in*: an gray level image.

Result

Returns a positive real value in [0..1].
(Use `pstatus` to get this value).

Examples

Computes the inter-region contrast measure for a simple binarization segmentation process:

```
pbinarization 80 1e30 tangram.pan i1.pan
plabeling 8 i1.pan i2.pan
paddcst 1 i2.pan i3.pan (label also the background)
pinterregioncontrast i3.pan tangram.pan
pstatus
```

See also

Evaluation, pintraregionuniformity

C++ prototype

```
Errc PInterRegionContrast( const Reg2d &rg_in, const Img2duc &im_in
);
```

Version française

Calcul du critère de qualité basé sur le contraste inter-régions.

Reference

[*] M. D. Levine and A. M. Nazif, "Dynamic measurement of computer generated image segmentations", *IEEE Trans. PAMI*, 7(2): 155-164, 1985.

Author: Régis Clouard

pintraregionuniformity

Computes the goodness measure based on intra-region uniformity.

Synopsis

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

Description

pintraregionuniformity computes the intra-region uniformity criterion as defined by M. Levine & A. Nazif[*]. This criterion relies on the idea that a good segmentation has a strong intra region uniformity. Uniformity is calculated from the inner contrast of the regions.

This criterion is near 1 when regions are homogeneous and near 0 when regions are heterogeneous.

The intra-region uniformity is computed as follows:

$$1 - \sum_R \left[\sum_s \left(ims[s] - mean(ims) \right)^2 \right]$$

where s is a site (pixel) of a region.

Caution: Regions with label=0 are not considered for computing.

Inputs

- *rg_in*: a region map.
- *im_in*: a gray level image.

Result

Returns the measure between [0..1].

(Use `pstatus` to get this value).

Examples

Computes the intra-region uniformity measure for a simple binarization segmentation process:

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

See also

Evaluation, pinterregioncontrast

C++ prototype

```
Errc PIntraRegionUniformity( const Reg2d &rg_in, const Img2duc
&im_in );
```

Version française

Calcul du critère de qualité basé sur une mesure d’uniformité des régions.

Reference

[*] M. D. Levine and A. M. Nazif, "Dynamic measurement of computer generated image segmentations", *IEEE Trans. PAMI*, 7(2): 155-164, 1985.

Author: Régis Clouard

pinverse

Performs binary inversion of image or graph and inversion of region map labels.

Synopsis

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

Description

pinverse performs a bitwise inversion of the values of the input *im_in*.

For image, inversion is applied on each pixel:

```
pixel = (max(image)+min(image))-pixel.
```

For color or multispectral image, the "inversion" operator is computed separately on each band.

For graph, inversion is applied on each node value:

```
val = (max(graph)-min(graph)-val.
```

For region map, inversion is applied on each label different from 0:

```
label 0 -> 0;
label 1 -> higher label value;
label 2 -> higher label value-1;
...
label higher label value -> 1;
```

Inputs

- *im_in*: an image, a graph or a region map.

Outputs

- *im_out*: an object of the same type as im_in.

Result

Returns SUCCESS or FAILURE.

Examples

```
pinverse tangram.pan a.pan
```

See also

logic

C++ prototype

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

Version française

Inversion binaire d’image ou de graphe, et inversion des numéros de labels de carte de régions.

Author: Régis Clouard

pkmeans

Performs K-means clustering on a set of objects.

Synopsis

pkmeans *attr_in attr_out k maxiter* [*col_in*|-] [*col_out*|-]

Description

pkmeans classifies a given set of objects into *K* clusters from their features. The object features are specified into *col_in* as a set of vectors `attr_in.1`, `attr_in.2`, ..., `attr_in.n`.

K-means is a partitioning method for a group of n objects into *k* clusters which uses the following steps:

1. Place *k* points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the distance to be minimized can be calculated.

The distance measure between an object i and the cluster center Cj uses the euclidean distance:

$$D_{ij} = \left[\sum_{d=1;n} (xid - Cjd)^2 \right]^{1/2}$$

where xid is the feature d for the object i and cjd is the feature d for the centroid Cj.

Parameters

- *attr_in* is the base name of the feature vector. The vectors are named `attr_in.1`, `attr_in.2`, ..., `attr_in.n` in the input collection. The item j of the array `attr_in.i` contains the (i)th feature of the (j+1)th object. They are Double arrays.
- *attr_out* is the name of the output array. Each item i of the array contains the number of the cluster from which the (i)th object is assigned. *attr_out* is an array of unsigned longs where attr_out[i] specifies the cluster number for the object i.
- *k* is the number of desired cluster.
- *maxiter* is the maximum number of iteration (in case of divergence).

Inputs

- *col_in*: a collection which contains the object features.

Outputs

- *col_out*: a collection which contains the assignment vector (object -> cluster).

Result

Returns SUCCESS or FAILURE.

Examples

Segments the tangram.pan image thanks to a K-means clustering of the pixels based on mean and variance features:

```
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
```

See also

Classification

C++ prototype

```
Errc PKmeans( const std::string &a_in, const Collection &c_in, const
std::string &a_out, Collection &c_out, int k, int max );
```

Version française

Classification automatique selon les K-moyennes.

Author: Alexandre Duret-Lutz

pknn

Performs KI-nearest neighbors clustering on a set of objects.

Synopsis

pknn *attr_base attr_in attr_out k* [*col_base*|-] [*col_in*|-] [*col_out*|-]

Description

pknn is a partitioning method for a group of n objects into *k* clusters. The classifier works based on minimum distance from the query instance to the training samples to determine the K-nearest neighbors. After we gather K nearest neighbors, we take simple majority of these K-nearest neighbors to be the prediction of the query instance.

The distance measure between two objects xi and xj uses the euclidean distance:

$$D_{ij} = [\text{SUM}_{\{d=1;n\}} (xid - xjd)^2]^{1/2}$$

where xid is the feature d for the object i and xjd is the feature d for the object xjd.

Parameters

- *attr_base* is the base name of the feature vector of the classified objects. The vectors are named `attr_base.1`, `attr_base.2`,..., `attr_base.n` in the input collection. The item j of the array `attr_in.i` contains the (i)th feature of the (j+1)th object. They are Double arrays. If the array `attr_base.C` is present then it contains the cluster number of each objects. Otherwise the ith object falls into the cluster i.
- *attr_in* is the base name of the feature vector of the objects to be classified. The vectors are named `attr_in.1`, `attr_in.2`, ..., `attr_in.n` in the input collection. The item j of the array `attr_in.i` contains the (i)th feature of the (j+1)th object. They are Double arrays.
- *attr_out* is the name of the output array. Each item i of the array contains the number of the cluster from which the (i)th object is assigned. *attr_out* is an array of unsigned longs where attr_out[i] specifies the cluster number for the object i.
- *k* is the number of desired cluster.

Inputs

- *col_base*: a collection which contains the feature vector of the classified objects.
- *col_in*: a collection which contains the feature vector of the objects to be classified.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Classifies beans into the jellybean.pan image from sample of each bean stored in the directory ’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
```

See also

Classification

C++ prototype

```
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);
```

Version française

Classification selon les k plus proches voisins.

Author: Alexandre Duret-Lutz

plab2lch

Converts Lab color image to LCH color image.

Synopsis

plab2lch [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

plab2lch converts a color image from the color space Lab (Luminancy, red/blue chrominancy, yellow/blue chrominancy) the to color space LCH (Light, Chroma, Hue). LCH is the perceptual version of HSL.

Inputs

- *im_in*: a Lab color image.

Outputs

- *im_out*: a LCH color image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts the color image a.pan from color space LAB to LCH:

```
plab2lch a.pan b.pan
```

See also

Color

C++ prototype

```
Errc PLAB2LCH( const Imc2duc &im_in, Imc2duc &im_out );
```

Version française

Changement d’espace couleur de Lab vers LCH.

Author: Olivier Lezoray

plabeling

Performs region labeling.

Synopsis

plabeling *connexity* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

plabeling builds regions from a set of connected pixels that have exactly the same pixel value. Two pixels p1 and p2 of the input image *im_in* are labeled with the same label value in the output region map *rg_out* if:

- p1 is a neighbor of p2 (according to the *connexity* value).
- im_in[p1] = im_in[p2].

A region is defined as a set of connected pixels with the same label. The label value is set randomly from 1 to the number of regions.

If *im_in* is already a region map, the **plabeling** relabels the region so as to minimize the number of used labels.

Notice: Pixels with label value = 0 are not considered as part of region. Thus, the region 0 does not exist.

Parameters

- *connexity* specifies the neighbor relation between pixels (4 or 8 for 2D; 6 or 26 for 3D).

Inputs

- *im_in*: a grayscale image, a region map or a graph.

Outputs

- *rg_out*: a region map or a graph.

Examples

Builds the skeleton by influence zones (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
```

Result

Returns the number of region or FAILURE.

See also

Segmentation, pboundarylabeling

C++ prototype

```
Errc PLabeling( const Img2duc &im_in, Reg2d &rg_out, int connexity
);
```

Version française

Etiquetage des régions homogènes d’une image.

Author: Régis Clouard

plabelmerging

Merges two regions from label value.

Synopsis

plabelmerging *label1 label2* [*rg_in*|-] [*gr_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

plabelmerging merges two regions specified by their label value. The regions of the output region map *rg_out* keep the same label than the regions in the input region map *rg_in*. The output graph *gr_out* is updated with the new node that results from the merging of the two nodes and also the list of neighbors. The new seed is located at the center of mass of the new region.

Parameters

- *label1* is the label of the first region in the input region map *rg_in*. It is an integer from 1 to maximum label value.
- *label2* is the label of the second region in the input region map *rg_in*. It is an integer from 1 to maximum label value.

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.

Outputs

- *rg_out*: a region map.
- *gr_out*: a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Segments the tangram.pan with a quadtree technique and then merges label 1 and 2:

```
puniformityquadtree 0.9 tangram.pan a.pan
prg2gr a.pan b.pan
plabelmerging  1 2 a.pan b.pan out1.pan out2.pan
```

See also

Segmentation

C++ prototype

```
Errc PLabelMerging( const Reg2d &rg_in, cconst Graph2d &gr_in, Reg2d
&rg_out Graph2d &gr_out, int label1, int label2 );
```

Version française

Fusion nominative de 2 régions.

Author: Régis Clouard

plabelselection

Selects region by label value.

Synopsis

plabelselection *label* [*rg_in*|-] [*rg_out*|-]

Description

plabelselection selects a region from its label value. The output region map *rg_out* is built with the unique region that has the specified *label* value.

The operator pthresholding can be used to select several regions from their label value.

Parameters

- *label* is the number of a region. If *label* =-1 or greater to the amount of labels then the greater label is used.

Inputs

- *rg_in*: a region map.

Outputs

- *rg_out*: a region map.

Result

Returns the number of selected regions.

Examples

Selects the region #10 from the region map rin.pan:

```
plabelselection 10 rin.pn rout.pan
```

See also

Region

C++ prototype

```
Errc PLabelSelection( const Reg3d &rg_in, Reg3d &rg_out, Long label
);
```

Version française

Sélection d’une région par son numéro de label.

Author: Régis Clouard

planczosrescale

Performs a rescaling of image using the Lanczos algorithm.

Synopsis

planczosrescale *zoomx zoomy zoomyz* [*im_in*|-] [*im_out*|-]

Description

planczosrescale uses a convolution kernel to interpolate the pixel of the input image *im_in* in order to calculate the pixel value of the output image *im_out*. The interpolation consists in weigthing the input pixels influence on the output pixels. The weights are relative to the position of the output pixels and are given by the Lanczos algorithm:

```
       | 1 if x=0
L(x) = | sinc(x).sin(x/a) if -a < x < a
       | 0 otherwise
```

For example, if the image is scaled by 3, then each output pixel is:

```
for i in [-3, 3]
  for j in [-3, 3]
      im_out[p] += L(i)*L(j)*im_in[p]
```

To rescale region map or graph, use the operator prescale.

Parameters

- *zoomx*, *zoomy*, *zoomz* are positive real values.
  - if a zoom factor is > 1 then the image is enlarged along the related axis.
  - if a zoom factor is < 1 then the image is shrunk along the related axis.

  (*zoomz* is ignored for 2D images but must be given).

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

- Enlarges the tangram.pan 2D image by a factor 2:

```
planczosrescale 2 2 0 tangram.pan a.pan
```

- Shrinks the tangram.pan 2D image by a factor 2:

```
planczosrescale 0.5 0.5 0 tangram.pan a.pan
```

See also

Transformation, plinearrescale, pbicubicrescale, pmitchellrescale, pbellrescale, prescale

C++ prototype

```
Errc PLanczosRescale( const Img2duc &im_in, Img2duc &im_out, const
float zoomy, const float zoomx );
```

Version française

Retaille d’une image par l’algorithme de Lanczos.

Author: Régis Clouard

plaplacian

Computes the Laplacian magnitude.

Synopsis

plaplacian *connexity* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

plaplacian computes the Laplacian of an image. The Laplacian is an approximation of the second derivative of an image.

This operator can be used to detect closed contours in an image. But it is very sensitive to noise.

The algorithm uses the convolution with the following kernel:

for 4-*connexity*

```
|+0 -1  +0|
|-1  +4 -1|
|+0 -1  +0|
```

for 8-*connexity*

```
|-1 -1- 1|
|-1  +8 -1|
|-1 -1 -1|
```

The output image *im_out* is of the same type as the input image *im_in*.

For Uchar images, the values are shifted by 127 so the 0 is 127 and value < 0 is supposed negative and values > 0 is supposed to be positive. The zero crossing detection needs to use operator pzerocross with parameter value 127.

For Signed Long images, the values are not shifted. The zero crossing detection needs to use operator pzerocross with parameter value 0.

Parameters

- *connexity* specifies the connexity neighbor relation {4 or 8}.

Inputs

- *im_in*: a 2D grayscale image.

Outputs

- *im_out*: an 2D grayscale image.

Result

Returns SUCCESS or FAILURE.

Exemples

Edge detection using LOG algorithm (Laplacian Of Gaussian)

```
pgaussfilter 1.2 tangram.pan a.pan
plaplacian 8  a.pan b.pan
pzerocross 8 127 b.pan out.pan
```

See also

Edge detection, pzerocross

C++ prototype

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

Version française

Approximation du Laplacian d’une image par convolution.

Author: Régis Clouard

pleafcutting

Performs leaf cutting.

Synopsis

pleafcutting [*-m mask*] [*gr_in*|-] [*gr_out*|-]

Description

pleafcutting deletes edges between nodes whose at least one vertex has only one neighbor (ie, is a leaf). The isolated vertex is however kept in the output graph *gr_out*.

Inputs

- *gr_in*: a graph.

Outputs

- *gr_out*: a graph.

Result

Returns the number of removed edges.

Examples

```
pleafcutting g1.pan g2.pan
```

See also

Graph

C++ prototype

```
Errc PLeafCutting( const Graph &gr_in, Graph &gr_out );
```

Version française

Suppression des feuilles d’un graphe.

Author: François Angot

plegendrepolynomialfitting

Approximates image to flat background using Legendre polynomial fitting.

Synopsis

plegendrepolynomialfitting *xorder yorder* [*im_in*|-] [*im_mk*|-] [*im_out*|-]

Description

plegendrepolynomialfitting converts an image content into a flat background image, using Legendre polynomial. It uses the orthogonality relation of the Legendre polynomials to expand an image as a double sum of those functions. The sum is then evaluated to produce an image that approximates a projection onto the space of polynomial images.

This operator removes all sorts of uneven illumination with a gradually change of background color raising or falling from any part of a border to the opposite border.

Parameters

- *xorder* is the x order [0..10].
- *yorder* is the y order [0..10].

Order values control the precision of the background fitting: the greater the order values, the closer to the input image content.

Inputs

- *im_in*: a 2D image.

Outputs

- *im_out*: a 2D image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

- Correction the illumination of the tangram image using background subtraction. The background is approximated by linear regression:

```
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
```

More examples

See also

Surface Fitting

C++ Prototype

```
Errc PLegendrePolynomialFitting( const Img2duc &im_in, Img2duc
&im_out, int yOrder, int yOrder );
```

Version française

Calcul de l'approximation du fond d'une image en utilisant une approximation par polynômes de Legendre.

Author: Régis Clouard

plineardilatation

Performs linear dilatation.

Synopsis

plineardilatation *orientation halfsize* [*im_in*|-] [*im_out*|-]

Description

plineardilatation dilates the points of stronger contrast with a linear element structuring of direction *orientation*. The size of the structuring element is of *halfsize* pixels on both sides of the central pixel.

Dilatation of point p corresponds to the operation: replace the central pixel by the maximum value of its neighbors:

```
dilatation(p) = MAX(neighbors specified by the structuring element).
```

For a binary image, dilatation dilates white areas.

For the region maps, dilatation dilates only regions that touch the background and region with the higher label are privileged.

For the color images, the lexicographic order is used: initially by using band X, in the event of equality by using the band Y then band Z.

Parameters

- *orientation* corresponds to the orientation of the structuring element according to the Freeman codes:

```
case of 2D: [0..3]
        1  2  3
        0     0
        3  2  1

case of 3D: [0..12]
z=-1:                z=0:           z=1:
         2  3  4        11 12  9         6  7  8
         1  0  5        10    10         5  0  1
         8  7  6         9 12 11         4  3  2
```

- *halfsize* is the half-size of the element structuring in a number of pixels. The line has a size of 2*halfsize+1.

Inputs

- *im_in*: an image or a region map.

Outputs

- *im_out*: an image (or a region map) of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Gets the tangram pieces boundaries that have at least 5 pixels high:

```
pgradient 1 tangram.pan i1.pan i2.pan
pbinarization 20 1e30 i1.pan i3.pan
plinearerosion 2 5 i3.pan i4.pan
plineardilatation 2 5 i4.pan out.pan
```

See also

Morphology, plinearerosion

C++ prototype

```
Errc PLinearDilatation( const Img2duc &im_in, Img2duc &im_out, int
orientation, int halsize );
```

Version française

Dilatation morphologique des points de plus fort contraste d’une image par une ligne.

Author: Régis Clouard

plinearerosion

Performs linear erosion.

Synopsis

plinearerosion *orientation halfsize* [*im_in*|-] [*im_out*|-]

Description

plinearerosion erodes the points of stronger contrast with a linear structuring element of direction *orientation*. The size of the structuring element is of *halfsize* pixels on both sides of the central pixel.

Erosion of a point p corresponds to the operation: replace the central pixel by the minimal value of its neighbors:

```
erosion(p) = MIN(neighbors specified by the structuring element).
```

For a binary image, erosion erodes white areas.

For the region maps, erosion adds pixels with label=0 (background) at the points of erosion.

For the color images, the lexicographic order is used: initially by using band X, in the event of equality by using the band Y then band Z.

Parameters

- *orientation* corresponds to the orientation of the structuring element according to the Freeman codes:

```
case of 2D: [0..3]
        1  2  3
        0     0
        3  2  1

case of 3D: [0..12]
z=-1:                z=0:           z=1:
         2  3  4        11 12  9         6  7  8
         1  0  5        10    10         5  0  1
         8  7  6         9 12 11         4  3  2
```

- *halfsize* is the half-size of the element structuring in a number of pixels. The line has a size of 2*halfsize+1.

Inputs

- *im_in*: an image or a region map.

Outputs

- *im_out*: an image (or a region map) of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Gets the tangram pieces boundaries that have at least 5 pixels high:

```
pgradient 1 tangram.pan i1.pan i2.pan
pbinarization 20 1e30 i1.pan i3.pan
plinearerosion 2 5 i3.pan i4.pan
plineardilatation 2 5 i4.pan out.pan
```

See also

Morphology, plineardilatation

C++ prototype

```
Errc PLinearErosion( const Img2duc &im_in, Img2duc &im_out, int
orientation, int halfsize );
```

Version française

Erosion morphologique des points de plus fort contraste d’une image par une ligne.

Author: Régis Clouard

plinearregression

Approximates image to flat background using linear regression.

Synopsis

plinearregression [*im_in*|-] [*im_out*|-]

Description

plinearregression converts an image content into a flat background image, using orthogonal linear regressions. This operator removes all sorts of uneven illumination with a gradually change of background color raising or falling from any part of a border to the opposite border. However, it cannot remove circular spotlight.

Inputs

- *im_in*: a 2D image.

Outputs

- *im_out*: a 2D image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

- Correction the illumination of the tangram image using background subtraction. The background is approximated by linear regression:

```
plinearregression tangram.pan a.pan
psub tangram.pan a.pan b.pan
pmeanvalue a.pan; mean=‘pstatus‘
paddcst $mean b.pan out.pan
```

More examples

See also

Surface Fitting

C++ Prototype

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

Version française

Calcul de l’approximation du fond d’une image en utilisant la regression linéaire.

Author: Régis Clouard

plinearrescale

Performs an affine rescaling of image using the linear interpolation.

Synopsis

plinearrescale *zoomx zoomy zoomyz* [*im_in*|-] [*im_out*|-]

Description

plinearrescale changes magnification of the input image by a factor *zoomx* along the x axis, *zoomy* along the y axis and *zoomz* along the z axis (for 3D images). The image is enlarged along an axis if the zoom factor is > 1 and is shrunk if the zoom factor is >0 and <1.

This version uses the bilinear interpolation. Bilinear interpolation considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixel:

```
sx = (x/zoomx)-||x/zoomx||
sy = (x/zoomy)-||y/zoomy||
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]);
```

The bilinear interpolation offers a good comprimise between time processing and result. For 2D image, a better result can be obtained with the bicubic interpolation but with greater processing time (see pbicubicrescale).

To rescale region map or graph, use the operator prescale.

Parameters

- *zoomx*, *zoomy*, *zoomz* are positive real values.

  if a zoom factor is > 1 then the image is enlarged along the related axis.

  if a zoom factor is < 1 then the image is shrunk along the related axis.
  (*zoomz* is ignored for 2D images but must be given).

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

- Enlarges the tangram.pan 2D image by a factor 2:

```
plinearrescale 2 2 0 tangram.pan a.pan
```

- Shrinks the tangram.pan 2D image by a factor 2:

```
plinearrescale 0.5 0.5 0 tangram.pan a.pan
```

See also

Transformation, pbicubicrescale, prescale

C++ prototype

```
Errc PlinearRescale( const Img2duc &im_in, Img2duc &im_out, const
float zoomy, const float zoomx );
```

Version française

Augmentation ou réduction de la taille d’une image par interpolation bilinéaire.

Author: Régis Clouard

plineartransform

Performs linear transform of the gray-levels.

Synopsis

plineartransform *inverse min max* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

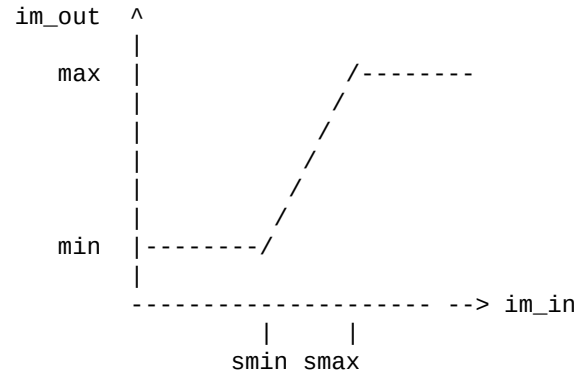
plineartransform expands or compresses gray-levels of the input image *im_in* using a linear transform of the gray-levels. The parameter *inverse* specifies whether the transform is positive or negative.

The effect of the positive transform is to stretch the gray-levels between the new bound [min..max]. The effect of the negative transform is to stretch the gray-levels between the new bounds [min,max] and to inverse all the gray-levels: max becomes min, min becomes max, etc.

The positive linear transform of pixel 'p' has the form:

```
im_out[p]=(c*(im_in[p]-smin)) + min;
c=(max-min) / (smax-smin)
```

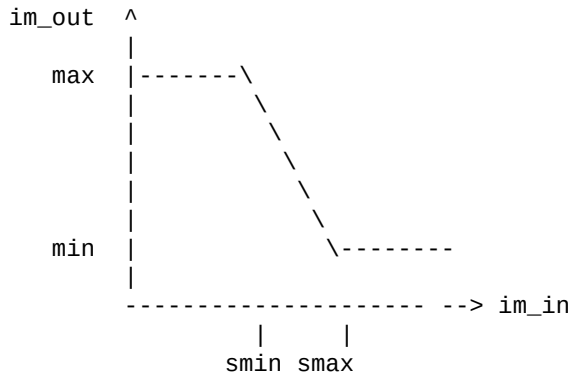
where smin and smax are the minimum and the maximum values of the input image, and c is a normalization factor for stretching output values between *min* and *max*.



The negative linear transform of pixel 'p' has the form:

```
im_out[p]=(c*(smax-ims[p])) + min;
c=(max-min) / (smax-smin)
```

where smin and smax are the minimum and the maximum values of the input image, and c is a normalization factor for stretching normalizing output values between min and max.



For color and multispectral images, the linear transform uses the vectorial approach: the min and max values are calculed from all the bands, and then each band is stretched with the same transform.

Parameters

- *inverse* is an integer in [0,1] that specifies whether the transform is positive (*inverse*=0) or negative (*inverse*=1).

- *min* and *max* specify the new bounds of the output pixel value. They are related to the type of the input image.
  Note: if *min* < *max* then min and max are set with the minimum and maximum values of the input image type; for example, 0 and 255 for Uchar images.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image with the same properties as im_in.

Result

Returns SUCCESS or FAILURE in case of invalid parameter values.

Examples

Applies a positive followed by a negative transform to create image b.pan. Because negative is the inverse transform of positive transform, b.pan is (almost) equal to tangram.pan (due to rounding error) and c.pan is (almost) null:

```
plineartransform 1 0 255 tangram.pan a.pan
plineartransform 0 0 255 a.pan b.pan
pdif a.pan b.pan c.pan
```

Applies a linear transform to create a.pan and uses the min and max values of the image type as new bound:

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

Piecewise linear transform: the input pixels between the bounds [0, 75] are compressed in the new bound [0..20] and those between the bounds [76, 255] are expanded in the new bounds [21, 255]:

```
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
```

See also

Lut transform, plogtransform, ppowerlawtransform

C++ prototype

```
Errc PLinearTransform( const Img2duc &im_in, const Img2duc &im_out,
int inverse, float min, float max );
```

Version française

Transformation linéaire des niveaux de gris.

Author: Régis Clouard

plipadd

Performs image addition according to the LIP model.

Synopsis

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

Description

plipadd computes the addition of the two input images *im_in1* and *im_in2*, according to the LIP model (Logarithmic Image Processing).

The LIP image addition is defined as:

```
im_out(x,y) = im_in1(x,y) + im_in2(x,y) - [ (im_in1(x,y).im_in2(x,y)) / M]
```

where M is the number of gray tones (eg, 256 for byte images)

The two inputs must be of the same type. The output image is of same type as the input images.

For color or multispectral image, the addition is computed separately on each band (marginal approach).

Inputs

- *im_in1*: an image.
- *im_in2*: an image.

Outputs

- *im_out*: an image of same type as input images.

Result

Returns SUCCESS or FAILURE.

Examples

Adds to images a.pan and b.pan, and result in result.pan:

```
plipadd a.pan b.pan result.pan
```

See also

Arithmetic

C++ prototype

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

Version française

Addition de 2 images selon le modèle LIP.

Author: Régis Clouard

plipmultcst

Performs scalar multiplication of image according to the LIP model.

Synopsis

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

Description

plipmultcst computes the scalar multiplication of an input image *im_in*, according to the LIP model (Logarithmic Image Processing).

The LIP scalar multiplication is defined as:

```
im_out(x,y) = M - M.[1-(im_in(x,y)/M)]^cst;
```

where M is the number of gray tones (eg, 256 for byte images).

The output file is of the same type as the input file.

For color or multispectral image, **plipmultcst** is computed separately on each band (marginal approach).

Parameters

- *cst* is a real value.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Divides the tangram.pan pixel values by 2:

```
plipmultcst 0.5 tangram.pan a.pan
```

See also

Arithmetic

C++ prototype

```
Errc PLipMultCst( const Img2duc &im_in, Img2duc &im_out, float cst
);
```

Version française

Multiplication d’une image par une constante selon le modèle LIP.

Author: Régis Clouard

plipsub

Performs image subtraction according to the LIP model.

Synopsis

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

Description

plipsub computes the subtraction of the two input images *im_in1* and *im_in2*, according to the LIP model (Logarithmic Image Processing).

The LIP image subtraction is defined as:

```
im_out(x,y) = M.(im_in1(x,y) - im_in2(x,y)) / ( M - im_in2(x,y) )
```

where M is the number of gray tones (eg, 256 for byte images).

The two inputs must be of the same type. The output image is of same type as the input images.

For color or multispectral image, the subtraction is computed separately on each band (marginal approach).

Inputs

- *im_in1*: an image.
- *im_in2*: an image.

Outputs

- *im_out*: an image of same type as the input images.

Result

Returns SUCCESS or FAILURE.

Examples

```
plipsub a.pan b.pan c.pan
```

See also

Arithmetic

C++ prototype

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

Version française

Soustraction de 2 images selon le modèle LIP.

Author: Régis Clouard

plocalextrema

Computes local extremum values of grayscale image.

Synopsis

plocalextrema *connexity* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

plocalmaxima builds the new image *im_out* with extremum pixels of the input image *im_in*. The value of the extremum pixel is the same as in the input image. All other pixels are set to 0.

A point is extremum if it is greater to at least one of two opposite neighbors and no lower than each of them:

```
im_in[y][x] > im_in[y-1][x] et im_in[y][x] >= im_in[y+1][x]
or
im_in[y][x] >= im_in[y-1][x] et im_in[y][x] > im_in[y+1][x]
```

Parameters

- *connexity* specifies the neighborhood relation (4, 8 in 2D) or (6, 26 in 3D). For graph, this parameter is ignored.

Inputs

- *im_in*: a grayscale image.

Outputs

- *im_out*: a grayscale image.

Result

Returns SUCCESS or FAILURE.

Examples

Detects the extrema in tangram.pan:

```
plocalextrema 8 tangram.pan a.pan
```

See also

Image Features Extraction

C++ prototype

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

Version française

Localisation des points constituant un extréma dans au moins direction.

Author: Régis Clouard

plocalmaxima

Computes local maximum values of grayscale image or graph.

Synopsis

plocalmaxima *connexity* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

plocalmaxima builds the new image *im_out* with the local maximum pixels of the input image *im_in*. The value of the maximum pixel is the same as in the input image. All other pixels are set to 0.

A point is a maximum if it is **strictly greater** than all its neighbors. Therefore, this operator cannot detect flat area. It might be necessary to use beforehand a smoothing operator (eg., pmeanfilter).

For graph, the maxima are computed from the node values.

Parameters

- *connexity* specifies the neighborhood relation (4, 8 in 2D) or (6, 26 in 3D). For graph, this parameter is ignored.

Inputs

- *im_in*: a grayscale image or a graph.

Outputs

- *im_out*: a grayscale image or a graph.

Result

Returns SUCCCESS or FAILURE.

Examples

Locates the local maxima in tangram.pan:

```
plocalmaxima 8 tangram.pan a.pan
```

See also

Image Features Extraction, plocalminima

C++ prototype

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

Version française

Localisation des points constituant un maximum local.

Author: Régis Clouard

plocalminima

Computes local minimum values of grayscale image or graph.

Synopsis

plocalminima *connexity* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

plocalminima builds the new image *im_out* with the local minimum pixels of the input image *im_in*. The value of the minimum pixel is the same as in the input image. All other pixels are set to 0.

A point is a minimum if it is **strictly lower** than all its neighbors. Therefore, this operator cannot detects flat area. It might be necessary to use beforehand a smoothing operator (eg., pmeanfilter).

For graph, the minima are computed from the node values.

Parameters

- *connexity* specifies the neighborhood relation (4, 8 in 2D) or (6, 26 in 3D). For graph, this parameter is ignored.

Inputs

- *im_in*: a grayscale image or a graph.

Outputs

- *im_out*: a grayscale image or a graph.

Examples

Locates the local minima in tangram.pan:

```
plocalminima 8 tangram.pan a.pan
```

Result

Returns SUCCESS or FAILURE.

See also

Image Features Extraction, plocalmaxima

C++ prototype

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

Version française

Localisation des points constituant un minimum local.

Author: Régis Clouard

plog

Computes natural logarithm of image or graph.

Synopsis

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

Description

plog computes the natural logarithm of the input *im_in*.

If *im_in* is an image then the new image *im_out* is built with the logarithm of each pixel. The problem of 0 is solved by using an epsilon. Thus the basis operation is as follows:

```
if (pixel(im_in) ==0 )
   pixel(im_out)=log(epsilon)
else
   pixel(im_out)=log(pixel(im_in))
```

The output image *im_out* is a always Float image.

For color or multispectral image, the logarithm is computed separately on each band.

If *im_in* is a graph then the new graph *im_out* is built with the natural logarithm of each node value.

Inputs

- *im_in*: an image or a graph.

Outputs

- *im_out*: a Float image or a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Computes the logarithm of the image tangram.pan :

```
plog tangram.pan a.pan
```

See also

Arithmetic

C++ prototype

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

Version française

Logarithme népérien d’une image or d’un graphe.

Author: Régis Clouard

plogtransform

Performs logarithmic and exponential transforms of the gray-levels.

Synopsis

plogtransform *inverse min max* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

plogtransform expands the gray-levels of the input image using a logarithmic or an exponential transform of the gray-levels. The parameter *inverse* specifies whether the transform is logarithmic or exponential.

The effect of the logarithmic transform is to map a narrow range of low gray-level values in the input image into a wider range of output levels. The inverse exponential transform is true for higher values of input levels.

The logarithmic transform of pixel 'p' has the form:

```
im_out[p]=(c*log(im_in[p]-smin+1.0)) + min;
c=(max-min) / (log(smax-smin+1.0))
```

where smin and smax are the minimum and the maximum values of the input image, and c is a normalization factor for stretching output values between *min* and *max*.

The exponential transform of pixel 'p' has the form:

```
im_out[p]=exp((im_in[p]-smin)/c) -1.0 + min;
c=(smax-smin) / (log(max-min+1.0))
```

where smin and smax are the minimum and the maximum values of the input image, and c is a normalization factor for stretching output values between min and max.

For color and multispectral images, the transform uses the vectorial approach: the min and max values are calculed from all the bands, and then each band is stretched with the same transform.

Parameters

- *inverse* is an integer in [0, 1] that specifies whether the transform is logarithmic (*inverse*=0) or exponential (*inverse*=1).

- *min* and *max* specify the bounds of the output pixel value. They are related to the type of the input image.
  Note: if *min* < *max* then min and max are set with the minimum and maximum values of the input image type; for example, 0 and 255 for Uchar images.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image with the same properties as im_in.

Result

Returns SUCCESS or FAILURE in case of invalid parameter values.

Examples

Applies a logarithmic transform followed by an exponential transform to create image b.pan. Because exp is the inverse transform of log, b.pan is (almost) equal to tangram.pan (due to rounding error).

```
plogtransform 0 0 255 tangram.pan a.pan
plogtransform 1 28 165 a.pan b.pan
```

Applies a logarithmic transform to create a.pan and uses the min and max values of the image type as new bounds.

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

See also

Lut transform, plineartransform, ppowerlawtransform

C++ prototype

```
Errc PLogTransform( const Img2duc &im_in, const Img2duc &im_out, int inverse, float min, float max );
```

Version française

Transformations des niveaux de gris par loi logarithmique ou exponentielle.

Author: Régis Clouard

pluv2lch

Converts L*u*v color image to LCH color image.

Synopsis

pluv2lch [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pluv2lch converts color image from color space Luv (Luminancy, chrominancy, chrominancy) the to color space LCH (Light, Chroma, Hue). LCH is perceptual version of HSL.

Inputs

- *im_in*: a L*u*v color image.

Outputs

- *im_out*: a LCH color image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts the color image a.pan from the color space Luv to LCH:

```
pluv2lch a.pan b.pan
```

See also

Color

C++ prototype

```
Errc PLAB2LCH( const Imc2duc &ims, Imc2duc &im_out);
```

Version française

Changement d’espace couleur Luv vers LCH.

Author: Olivier Lezoray

pmalikperonafiltering

Performs non linear diffusion smoothing.

Synopsis

pmalikperonafiltering *iterations edgethreshold* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pmalikperonafiltering performs filtering on the input image *im_in* from the Malik-Perona algorithm. It is based on the equation diffusion:

```
c=exp (-(|grad(im_in)|/K)^2)
```

where K = edgethreshold.

The image border (of size 1) is not considered for processing. The output image border is just a copy of the input image border.

Parameters

- *iterations* specifies the number of iterations required to solve the model equation. It is a positive integer. The number of iterations strongly depends on the size of the objects contained in the image.
- *edgethreshold* specifies edge threshold parameter. It is a positive integer. Edge with gradient magnitude > 10 are preserved whereas other are smoothed. A typical value is 10.

Inputs

- *im_in*: a grayscale image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Applies the Malik-Perona filter to tangram.pan. It preserves edge with magnitude > 10 and performs 40 iterations:

```
pmalikperonafiltering 40 10 tangram.pan out.pan
```

See also

Filtering

C++ prototype

```
Errc PMalikPeronaFiltering( const Img2duc &im_in, Img2duc &im_out,
int iterations, int edgethreshold );
```

Version française

Lissage d’une image par diffusion non linéaire selon l’algorithme de Malik-Peronna.

Author: Sophie Sch&uum;pp

pman

Finds and displays reference manual pages.

Synopsis

pman [*-M path*] *operator_name ...*
pman [*-M path*] *-k keyword ...*

Description

pman displays information about Pandore operators from the reference manuals. It displays complete manual pages about an operator selected by name, or one-line summaries for several operators selected by keyword.

Manual pages must be HTML files and must be included in a subdirectory named `operatorPxx` where xxx is a number. For example `/usr/local/operatorP0`.

Parameters

- *-k keyword* ... prints out one-line summaries from the table of contents of Pandore.
- *-M directory* specifies an alternate search path for manual path. Manual pages are searched for the directory `directory/operatorsPxxx` and keywords are searched for the file `directory/operatorPxx.html`.

Inputs

- *operator_name*: the name of a Pandore operator.

Result

No result value.

Examples

Displays manual page about the operator "pman".

```
pman pman
```

Displays all operators that deal with word "segmentation".

```
pman -k segmentation
```

Displays manual page about the local operator "pfoo" located in the "/usr/local/myoperators/doc" directory.

```
pman -M /usr/local/myoperators/doc pfoo
```

See also

Information

Version française

Affichage en ligne de la documentation associée à un opérateur.

Author: Régis Clouard

pmask

Performs masking of image, graph or region map by an image or a region map.

Synopsis

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

Description

pmask applies the mask *im_in2* on the input *im_in1*.

For image, masking is applied on each pixel:

```
if pixel(im_in2)
   pixel(im_out) = pixel(im_in1)
else
   pixel(im_out) = 0
```

For color or multispectral image, the "mask" operator is computed separately on each band.

For graph, masking is applied on each node value.

For region map, masking is applied on label. All masked label are set to 0 in the result, and other labels are copied onto the output *im_out*.

Inputs

- *im_in1*: an image, a graph or a region map.
- *im_in2*: a gray level image or a region map.

Outputs

- *im_out*: an object of the same type as *im_in1*.

Result

Returns SUCCESS or FAILURE.

Examples

```
pmask a.pan b.pan c.pan
```

See also

logic

C++ prototype

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

Version française

Masquage d’un objet par une image ou une carte de régions.

Author: Régis Clouard

pmassthresholding

Performs thresholding on image using gray level percent.

Synopsis

pmassthresholding *ratio* [*-m mask* [*im_in*|-] [*im_out*|-]

Description

pmassthresholding sets to 0 every pixel value that is lower to *ratio* percent of the total gray levels.

The threshold value is determined as the gray level that splits the image in two class:

- *ratio*% pixels with value < threshold;
- 100-*ratio* pixels with value ≥ threshold.

Pixel with value lower or equal to the threshold are set to 0 in the output image, and pixels with value greater than the threshold are kept in the output image.

Parameters

- *ratio* is a real from [0..100] that corresponds to the percentage of pixel in the first cluster.

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).

Outputs

- *im_out*: an image of the same type as *im_in*.

Result

Returns the threshold value.

Examples

Discards the background and keeps the tangram pieces from the tangram.pan image:

```
pmassthresholding 86 tangram.pan out.pan
```

See also

Thresholding

C++ prototype

```
Errc PMassThresholding( const Img2duc &im_in, Img2duc &im_out, Float
ratio );
```

Version française

Seuillage d’une image basé sur le pourcentage de niveaux de gris.

Author: Régis Clouard

pmax

Performs maximum values between image or a graph.

Synopsis

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

Description

pmax computes the maximum values between inputs *im_in1* and *im_in2*.

If *im_in1* and *im_in2* are images then the new image *im_out* is built with the maximum between pixel values of the two input images:

```
if (pixel(im_in1) < pixel(im_in2))
then pixel(im_out) = pixel(im_in2)
else pixel(im_out) = pixel(im_in1)
```

The input image *im_in1* and *im_in2* must be of the same type, and the output image *im_out* is of the same type as the two input images.

For color or multispectral image, the maximum is computed separately on each band.

If *im_in* is a graph then the new graph *im_out* is built with the maximum of values of nodes with the same index.

Inputs

- *im_in1*: an image, a graph or a region map.
- *im_in2*: an image, a graph or a region map.

Outputs

- *im_out*: an object of the same type as the inputs.

Result

Returns SUCCESS or FAILURE.

Examples

Computes the maximum between pixels of the two images a.pan and b.pan:

```
pmax a.pan b.pan c.pan
```

See also

Arithmetic

C++ prototype

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

Version française

Maximum entre valeurs d’images ou de graphes.

Author: Régis Clouard

pmaximumselection

Selects regions from maximum grayscale value.

Synopsis

pmaximumselection *relation threshold* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

pmaximumselection selects regions from their maximum pixel value. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

The maximum value is calculated from the input image *im_in* and the region location is given in the input region map *rg_in*.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum maximum value.
- *threshold* is an integer defined in grayscale unit.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of selected regions.

Examples

Select regions with the maximum value > 50:

```
pmaximumselection 1 50 rin.pan a.pan rout.pan
```

See also

Region, pminimumselection

C++ prototype

```
Errc PMaximumSelection( const Reg2d &rg_in, Img2duc &im_in, Reg2d
&rg_out, int relation, Uchar threshold );
```

Version française

Sélection de régions sur leur valeur de maximum intérieur.

Author: Régis Clouard

pmaximumvalue

Measures global maximum value of image, region map or graph.

Synopsis

pmaximumvalue [*-m mask*] [*im_in*|-] [*col_out*|-]

Description

pmaximumvalue returns the maximum value in the input image or region map or graph *im_in*. if *im_in* is an image, the maximum value is the maximum pixel value on each band. If *im_in* is a region map, the maximum value is the maximum label value. If *im_in* is a graph, the maximum value is the maximum node value.

The maximum values for each band are stored in the collection *col_out*.

Inputs

- *im_in*: an image, a region map or a graph.

Outputs

- *col_out*: a collection of float values.

Result

Returns the maximum value (for the first band only). This value can be get using operator **pstatus**.

Examples

Measures the global maximum of the tangram.pan (Unix version):

```
pmaximumvalue tangram.pan col.pan
val=‘pstatus‘
echo "Maximum = $val"
```

Measures the global maximum of the tangram.pan (MsDos version):

```
pmaximumvalue tangram.pan col.pan
call pstatus
call pset val
echo Maximum = %val%
```

See also

Image Features Extraction

C++ prototype

```
Float PMaximumValue( const Img2duc &im_in, Collection & col_out );
```

Version française

Recherche de la valeur de pixel maximum dans une image (un graphe ou une carte de régions).

Author: Régis Clouard

pmaxprojection

Performs maximum value orthogonal projection along main axis.

Synopsis

pmaxprojection *axis* [*im_in*|-] [*im_out*|-]

Description

pmaxprojection builds a new image *im_out* with one dimension less than the input image *im_in* which stores the projection of the maximum value along the specified axis. If the input image is a 3D image, the output image is a 2D image where each pixel stores the maximum value along the orthogonal axis. If the input image is a 2D image, the output image is a 1D image where each pixel stores the maximum value along the orthogonal axis.

For example, the projection of the 2D image along the x axis builds the output image *im_out* of the size width(*im_in*) and each pixel is set with:

$$im\_out[x] = MAX_y(im\_in[y][x])$$

Parameters

- *axis* is an integer from [0..3] which specifies the projection axis:
  - 0: along X,
  - 1: along Y,
  - 2: along Z (3D only).

Inputs

- *im_in*: a 2D or 3D image.

Outputs

- *im_in*: a 2D or 1D image.

Result

Returns SUCCESS or FAILURE.

Examples

Projects the gray levels of the tangram.pan image along the x axis:

```
pmaxprojection 0 tangram.pan a.pan
```

See also

Transformation

C++ prototype

```
Errc PMaxProjection( const Img3duc &im_in, Img2duc &im_out, int axis
);
```

Version française

Projection orthogonale des valeurs maximales sur un axe d’une image.

Author: François Angot

pmcmfiltering

Performs Mean Curvature Motion filtering on image.

Synopsis

pmcmfiltering *iterations* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pmcmfiltering performs a Mean Curvature Motion filtering on the input image *im_in*. It is based on the diffusion equation:

```
d(im_in/dt) - Curv(im_in).|grad(im_in)|
```

where Curv(im-in) = div(grad(im_in) / |grad(im_in)|).

Concretely, the diffusion is made in the orthogonal direction with the gradient and thus it preserves the edges and smoothes the remainder of the image.

The image border (of size 1) is not considered for processing. The output image border is just a copy of the input image border.

Parameters

- *specifies the number of iterations required to solve the model equation. It is a positive integer. The number of iterations strongly depends on the size of the objects contained in the image.*

Inputs

- *im_in: an image.*

Outputs

- *im_out: an image of the same type as the input image.*

Result

Returns SUCCESS or FAILURE.

Examples

Applies the Mean Curvature Motion filter to tangram.pan:

```
pmcmfiltering 2 tangram.pan out.pan
```

See also

Filtering

C++ prototype

```
Errc PMcmFiltering( const Img2duc &im_in, Img2duc &im_out, int
iterations );
```

Version française

Lissage d’une image par filtre de courbure moyenne.

Author: Abderrahim Elmoataz

pmean

Performs average between images or graphs.

Synopsis

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

Description

pmean computes the average between the two inputs *im_in1* and *im_in2*.

If inputs are image, the average is done on each pixel values:

```
pixel(im_out) = ((pixel(im_in1) + pixel(im_in2))/2;
```

The two inputs must be of the same type.

For color or multispectral image, the average is computed separately on each band.

If *im_in1* and *im_in2* are graphs then the new graph *im_out* is built with the average of node values.

Inputs

- *im_in1*: an image or a graph.
- *im_in2*: an image or a graph.

Outputs

- *im_out*: an object of the same type as the inputs.

Result

Returns SUCCESS or FAILURE.

Examples

Computes the mean between pixels of the two images a.pan and b.pan:

```
pmean a.pan b.pan c.pan
```

See also

Arithmetic

C++ prototype

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

Version française

Moyennage entre images ou graphes.

Author: Régis Clouard

pmeanaggregation

Performs pixel aggregation based on mean criterion.

Synopsis

pmeanaggregation *connexity threshold* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

pmeanaggregation builds a new region map from aggregation of pixels to regions of the input region map *rg_in*. A pixel p is aggregated to a region R if:

- p is connected to the region R according to the specified *connexity*;
- |mean(R)-mean(R+im_in[p])| ≤ threshold

The mean of the region is not updated with the new pixel to avoid moving away too much from the initial situation. One prefer iterative executions of the operator to update the inner mean. For example, operator can be iterated until pstatus returns 0.

The output region map *rg_out* has the same number of labels than the input region map.

Parameters

- *connexity* specifies the neighbor relation between pixels (4 or 8 for 2D; 6 or 26 for 3D).
- *threshold* specifies the maximum mean value to decide to aggregate a pixel to the region. Values are from the gray scale of the input image.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of aggregation or FAILURE.

Examples

Aggregates pixels to tangram pieces:

```
pbinarization 96 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pmeanaggregation 8 45 b.pan tangram.pan out.pan
```

See also

Segmentation

C++ prototype

```
Errc PMeanAggregation( const Reg2d &rg_in, Img2duc &im_in, Reg2d
&rg_out, int connexity, Uchar threshold );
```

Version française

Croissance des régions d’une carte selon la moyenne intérieure.

Author: Régis Clouard

pmeanfiltering

Performs mean filtering on image or graph.

Synopsis

pmeanfiltering *halfsize* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pmeanfiltering applies a mean filter to the input image *im_in*. Each pixel of the input image is replaced by the mean of its neighbors. The neighborhood size is defined by the parameter *halfsize*. For example, **pmeanfiltering** with *halfsize*=1 corresponds to a 3x3 filter and the central pixel is replaced by the convolution with the follows kernel:

```
1/9  1/9  1/9
1/9  1/9  1/9
1/9  1/9  1/9
```

Each neighbor is multiplied by 1/9 and the central pixel is replaced by the sum.

The image border (of size halfsize) is not considered for processing. The output image border is just a copy of the input image border.

For graph, **pmeanfiltering** is applied on node values.

Parameters

- *halfsize* specifies the size of the neighborhood. It is a positive integer. A *halfsize*=1 corresponds to a 3x3 neighborhood. The greater is halfsize, the stronger is the filtering.
  This parameter is ignored for graph but must be given.

Inputs

- *im_in*: an image or a graph.

Outputs

- *im_out*: an image of the same type as the input image or a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Applies a 5x5 mean filter to tangram.pan:

```
pmeanfiltering 2 tangram.pan out.pan
```

See also

Filtering

C++ prototype

```
Errc PMeanFiltering( const Img2duc &im_in, Img2duc &im_out, int
halfsize );
```

Version française

Lissage d’une image par un filtre moyenneur linéaire.

Author: Régis Clouard

pmeanmerging

Performs priority region merge based on mean criterion.

Synopsis

pmeanmerging *iterations threshold* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

pmeanmerging merges connected regions of the input image *rg_in* if the difference between the mean value of the region is lower than the specified *threshold*.

Two regions are connected if there exists a link between the related nodes in the input graph *gr_in*.

The principle of the algorithm is as follows:

- For each region of the input region map *rg_in*:
- If the difference between the criterion value of the two connected regions ≤ *threshold* then merge them into one region.

The algorithm uses the priority merge that consists in merging regions with the lower difference.

The output region map *reg_out* defines the new regions and the output graph *gr_out* defines the new relationship between regions.

Parameters

- *iterations* specifies the number of merges to perform. If *number* = -1 then all possible merges are done.
- *threshold* specifies the maximum difference allowed between two regions to decide to merge them. Values are from the gray scale of the input image.

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.
- *gr_out*: a graph.

Result

Returns the number of merges.

Examples

Merges regions yielded by a quadtree splitting process:

```
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
```

See also

Segmentation

C++ prototype

```
Errc PMeanMerging( const Reg2d &rg_in, const Graph2d &gr_in, const
Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, long iterations,
float threshold );
```

Version française

Fusion prioritaire de régions selon la différence de moyennes intérieures.

Author: Laurent Quesnel

pmeanprojection

Performs mean value orthogonal projection along main axis.

Synopsis

pmeanprojection *axis* [*im_in*|-] [*im_out*|-]

Description

pmeanprojection builds a new image *im_out* with one dimension less than the input image *im_in* which stores the projection of the mean value along the specified axis. If the input image is a 3D image, the output image is a 2D image where each pixel stores the mean value along the orthogonal axis. If the input image is a 2D image, the output image is a 1D image where each pixel stores the mean value along the orthogonal axis.

For example, the projection of the 2D image along the x axis builds the output image *im_out* of the size width(*im_in*) and each pixel is set with:

$$im\_out[x] = MEAN_y(im\_in[y][x])$$

Parameters

- *axis* is an integer from [0..3] which specifies the projection axis:
  - 0: along X,
  - 1: along Y,
  - 2: along Z (3D only).

Inputs

- *im_in*: a 2D or 3D image.

Outputs

- *im_in*: a 2D or 1D image.

Result

Returns SUCCESS or FAILURE.

Examples

Projects the gray levels of the tangram.pan image along the x axis:

```
pmeanprojection 0 tangram.pan a.pan
```

See also

Transformation

C++ prototype

```
Errc PMeanProjection( const Img3duc &im_in, Img2duc &im_out, int
axis );
```

Version française

Projection orthogonale des valeurs moyennes sur un axe d’une image.

Author: François Angot

pmeanselection

Selects regions from mean grayscale value.

Synopsis

pmeansselection *relation threshold* [*-m mask*] [*rg_in*|-] [*im_in*|-][*rg_out*|-]

Description

pmeanselection selects regions from their mean value. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

The mean value mi of the region i is calculated as follows:

```
mi= SUM(im_in[p] / p in Ri) / N
```

where N is the number of pixels in the region.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is an integer defined in grayscale unit.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of selected regions.

Examples

Selects regions with a mean value > 50:

```
pmeanselection 1 50 rin.pan a.pan rout.pan
```

See also

Region

C++ prototype

```
Errc PMeanSelection( const Reg2d &rg_in, const Img2duc &im_in, Reg2d &rg_out, int relation, Uchar threshold );
```

Version française

Sélection de régions sur leur valeur de moyenne intérieure.

Author: Régis Clouard

pmeanshiftsegmentation

Performs pixel classification on image using mean shift algorithm.

Synopsis

pmeanshiftsegmentation *spatial-bandwidth range-bandwidth minimum-region-area accuracy-level* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

The **pmeanshiftsegmentation** classifies the input image pixels into regions using color homogeneity. It is a procedure for locating the maxima of a density function given discrete data sampled from that function. The algorithm is based on the detection of the modes of this density function.

The parameters controls the classification process. The *spatial-bandwidth* controls the size of the search window when computing mean shift. The *range-bandwidth* controls the color vicinity for homogeneous regions. The more the spatial-bandwidth and the range-bandwith, the less number of regions.

The result is the region map *rg_out*.

Parameters

- *spatial-bandwidth* specifies a spatial search window of size (2r+1)*(2r+1) during the mean shift computation, where r is the spatial bandwidth. It's an integer value greater than zero.
- *range-bandwidth* specifies the bandwidth of the search window in the range subspace during the computation of mean shift. It's a real value greater than zero (should not be high, generally <10).
- *minimum-region-area* specifies the minimum allowable region area (in pixels) contained in the segmented image. It's an integer greater than zero given in pixels.
- *accuracy-level* determines the accuracy level used when computing mean shift. Value is an integer in [0..2], where 2 is the maximum accuracy level. However the more is the accuracy level, the less is speed level. 2 gives the most accurate results. Note that the size of the *spatial-bandwidth* parameter also infers on the speed.

Inputs

- *im_in*: a 2D image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of regions or FAILURE.

Examples

Segments the tangram pieces:

```
pmeanshiftsegmentation 10 20 100 0 tangram.pan a.pan;
```

See also

Segmentation

C++ prototype

```
Errc PMeanShiftSegmentation( const Imc2duc &ims, Reg2d &rgd, int
spatialBandwidth, float rangeBandwidth, int minimumRegionArea, int
accuracyLevel );
```

Version française

Classification des pixels d’une image par l’algorithme Mean-Shift.

Reference

D. Comanicu, P. Meer: "Mean shift: A robust approach toward feature space analysis." IEEE Trans. Pattern Anal. Machine Intell., 24, 603-619, May 2002

Author: Régis Clouard

pmeanvalue

Measures global mean value of grayscale image or graph.

Synopsis

pmeanvalue [*im_in*|-] [*col_out*|-]

Description

pmeanvalue measured the mean value of the input object *im_in* for *non null values only*. If *im_in* is a grayscale image, the mean value is calculated from the pixel value. If *im_in* is a graph, the mean value is calculated from the node values.

The mean value is calculated as follows:

```
mean = SUM (im_in(x,y)) / N; if im_in(x,y) != 0.
```

where N is the number of pixels (or nodes).

The mean values for each band are stored in the collection *col_out*.

Note : This operator is not maskable.

Inputs

- *im_in*: an image or a graph.

Outputs

- *col_out*: a collection of float values.

Result

Returns the global mean value (for the first band only). This value can be get using operator **pstatus**.

Examples

Measures the global mean the tangram.pan (Unix version):

```
pmeanvalue tangram.pan col.pan
var=‘pstatus‘
echo "Mean = $val"
```

Measures the global mean of the tangram.pan (MsDos version):

```
pmeanvalue tangram.pan col.pan
call pstatus
call pset var
echo Mean = %val%
```

See also

Image Features Extraction

C++ Prototype

```
Float PMeanValue( const Img2duc &im_in, Collection & col_out );
```

Version française

Calcul du niveau de gris moyen d’une image.

Author: Régis Clouard

pmedianfiltering

Performs median filtering on image.

Synopsis

pmedianfiltering *halfsize* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pmedianfiltering applies a median filter to the input image *im_in*. Each pixel of the input image is replaced by the median value of its neighbors. The neighborhood size is defined by the parameter *halfsize*.

It is an rank filtering which can be used to remove impulse and exponential noise. It remove small details while preserving edge of type "step". However, this filter can affect the image geometry. For example, angle tends to be rounded, and edge of type roof and "peak" tends to be removed.

Parameters

- *halfsize* specifies the size of the neighborhood. It is a positive integer. A *halfsize*=1 corresponds to a 3x3 neighborhood. This parameter is ignored for graph but must be given. The greater is halfsize, the stronger is the filtering.

Inputs

- *im_in*: a grayscale image.

Inputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Applies a 5x5 median filter to tangram.pan:

```
pmedianfiltering 2 tangram.pan out.pan
```

See also

Filtering

C++ prototype

```
Errc PMedianFiltering( const Img2duc &im_in, Img2duc &im_out, int
halfsize );
```

Version française

Lissage d’une image par médian standard séparable.

Author: Julien Robiaille

pmedianvalue

Measures median value of grayscale image.

Synopsis

pmedianvalue [*im_in*|-] [*col_out*|-]

Description

pmedianvalue returns the median value of the input image *im_in*.

To calculate the median value, the pixel are sorted in increasing order. If the number of pixels is odd then the median value is the middle value. If the number of pixels is even then the median value is the mean value between the two pixel values around the center.

The median values for each band are stored in the collection *col_out*.

Inputs

- *im_in* : an image or a graph.

Outputs

- *col_out*: a collection of float values.

Result

Returns the median value (for the first band only). This value can be get using operator **pstatus**.

Examples

Measures the median the tangram.pan (Unix version):

```
pmedianvalue tangram.pan col.pan
var=‘pstatus‘
echo "Median = $val"
```

Measures the median of the tangram.pan (MsDos version):

```
pmedianvalue tangram.pan col.pan
call pstatus
call pset var
echo Median = %val%
```

See also

Image Features Extraction

C++ prototype

```
Float PMedianeValue( const Img2dsf &im_in, Collection & col_out );
```

Version française

Recherche de la valeur médiane d’une image.

Author: Jalal Fadili

pmergeimages

Merges 4 images into one image.

Synopsis

pmergeimages [*im_in1*|-] [*im_in2*|-] [*im_in3*|-] [*im_in4*|-] [*im_out*| -]

Description

pmergeimages merges the four input images *im_in1*, *im_in2*, *im_in3*, *im_in4* to build the output image *im_out*. They are merged as follows:

```
[im_in1][im_in2]
[im_in3][im_in4]
```

The input images are supposed to be of the same type. They size must be compatible:

- *im_in1* width = *im_in3* width;
- *im_in2* width = *im_in4* width;
- *im_in1* height = *im_in2* height;
- *im_in3* height = *im_in4* height;

The size of the output image is (*im_in1* + *im_in2*) width and (*im_in1* + *im_in3*) height.

Inputs

- *im_in1*, *im_in2*, *im_in3*, *im_in4*: 2D images of the same type and with compatible size.

Inputs

- *im_out*: a 2D image.

Result

Returns SUCCESS or FAILURE.

Examples

Builds a synthetic image to illustrate the Gibbs phenomenon in wavelets analysis.

```
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
```

See also

Utility, psplitimage

C++ prototype

```
Errc PPsplitImage( const Img2dsd &im_in1, const Img2dsd &im_in2,
const Img2dsd &im_in3, const Img2dsd &im_in4, Img2dsd &im_out );
```

Version française

Regroupement de 4 sous-images en une seule.

Author: Ludovic Soltys

pmin

Computes minimum values between image or graph.

Synopsis

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

Description

pmin computes the minimum values between inputs *im_in1* and *im_in2*.

If *im_in1* and *im_in2* are images then the new image *im_out* is built with the minimum between pixel values of the two input images:

```
if (pixel(im_in1) > pixel(im_in2))
then pixel(im_out) = pixel(im_in2)
else pixel(im_out) = pixel(im_in1)
```

The input image *im_in1* and *im_in2* must be of the same type, and the output image *im_out* is of the same type as the two input images.

For color or multispectral image, the minimum is computed separately on each band.

If *im_in* is a graph then the new graph *im_out* is built with the minimum of values of nodes with the same index.

Inputs

- *im_in1*: an image or a graph.
- *im_in2*: an image or a graph.

Outputs

- *im_out*: an object of the same type as the inputs.

Result

Returns SUCCESS or FAILURE.

Examples

Computes the minimum between pixels of the two images a.pan and b.pan:

```
pmin a.pan b.pan c.pan
```

See also

Arithmetic

C++ prototype

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

Version française

Minimum entre valeurs d’images ou de graphes.

Author: Régis Clouard

pminimumselection

Selects regions from minimum grayscale value.

Synopsis

pminimumselection *relation threshold* [*-m mask*] [*rg_in*|-] [*im_in*|-][*rg_out*|-]

Description

pminimumselection selects regions from their minimum pixel value. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

The minimum value is calculated from the input image *im_in* and the region location is given in the input region map *rg_in*.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum minimum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum minimum value.
- *threshold* is an integer defined in grayscale unit.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of selected regions.

Examples

Selects the region(s) with the most minimum grayscale:

```
pminimumselection -3 0 rin.pan a.pan rout.pan
```

See also

Region, pmaximumselection

C++ prototype

```
Errc PMinimumSelection( const Reg2d &rg_in, Img2duc &im_in, Reg2d
&rg_out, int relation, Uchar threshold );
```

Version française

Sélection de régions sur leur valeur de minimum intérieur.

Author: Régis Clouard

pminimumvalue

Measures global minimum value of image, region map or graph.

Synopsis

pminimumvalue [*-m mask*] [*im_in*|-] [*col_out*|-]

Description

pminimumvalue returns the minimum value in the input object *im_in*. if *im_in* is an image, the minimum value is the minimum pixel value on each band. If *im_in* is a region map, the minimum value is the minimum label value. If *im_in* is a graph, the minimum value is the minimum node value.

The minimum values for each band are stored in the collection *col_out*.

Inputs

- *im_in*: an image, a region map or a graph.

Outputs

- *col_out*: a collection of float values.

Result

Returns the minimum value (for the first band only). This value can be get using operator **pstatus**.

Examples

Measures the global minimum of the tangram.pan (Unix version):

```
pminimumvalue tangram.pan col.pan
var=‘pstatus‘
echo "Minimum = $val"
```

Measures the global minimum of the tangram.pan (MsDos version):

```
pminimumvalue tangram.pan col.pan
call pstatus
call pset var
echo Minimum = %val%
```

See also

Image Features Extraction

C++ prototype

```
Float PMinimumValue( const Img2duc &im_in, Collection & col_out );
```

Version française

Recherche de la valeur de pixel minimum dans une image (un graphe ou une carte de régions).

Author: Régis Clouard

pmitchellrescale

Performs a rescaling of image using the Mitchell algorithm.

Synopsis

pmitchellrescale *zoomx zoomy zoomyz* [*im_in*|-] [*im_out*|-]

Description

pmitchellrescale uses a convolution kernel to interpolate the pixel of the input image *im_in* in order to calculate the pixel value of the output image *im_out*. The interpolation consists in weigthing the input pixels influence on the output pixels. The weights are relative to the position of the output pixels and are given by the Mitchell algorithm:

```
Let 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 if -1
```

pmodulus

Computes modulus between 2 images.

Synopsis

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

Description

pmodulus computes the modulus between 2 images considered as a unique complex image.

The modulus between 2 images is calculated as follows:

```
pixel(im_out) = sqrt(pixel(im_in1)*pixel(im_in1) + pixel(im_in2)*pixel(im_in2))
```

Inputs

- *im_in1*: a grayscale or a color image.
- *im_in2*: a grayscale or a color image.

Outputs

- *im_out*: a float image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts an image of a square from the spatial domain to the frequency domain and reciprocally:

```
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
```

See also

Frequency, pphase

C++ Prototype

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

Version française

Calcul du module entre deux images.

Author: Régis Clouard

pmse

Computes the Mean Square Error.

Synopsis

pmse [*im_in1*|-] [*im_in2*|-]

Description

pmse measures the Mean Squared Error (MSE) between the initial image *im_in1* and the restored or enhanced image *im_in2*.
A lower value for MSE means lesser error. However, the MSE depends on the maximum values of the input images. For example, a MSE=100.0 for Uchar image is very high whilst a MSE=100.0 for Long images with value in [0...65535] is very low.

MSE is defined as follows:

```
MSE=1/N * sum { (im_in1-im_in2)^2 }
```

where N is the total number of pixel in the input image *im_in1*.

Input images *im_in1* and *im_in2* must have the same dimensions and the same type.

For color images or multispectral images, the definition of MSE is the same except that the sum over all squared value differences is also divided by the number of bands (ie. 3 for the color images).

Inputs

- *im_in1*: an image.
- *im_in2*: an image (restored or enhanced version of im_in1).

Result

Returns the value as a positive real value.
(Use `pstatus` to get this value).

Examples

Computes the MSE for the meanfilter smoothing operator:

```
pmeanfiltering 2 tangram.pan i1.pan
pmse tangram.pan i1.pan
pstatus
```

See Also

Evaluation, psnr, ppsnr

C++ prototype

```
Errc PMSE( const Img2duc &im_in1, const Img2duc &im_in2 );
```

Version française

Calcul de l’Erreur Quadique Moyenne (Mean Square Error).

Author: Régis Clouard

pmst

Builds the Minimum Spanning Tree of graph.

Synopsis

pmst [*-m mask*] [*gr_in*|-] [*gr_out*|-]

Description

pmst builds the Minimum Spanning Tree of the input graph *gr_in*.

A spanning tree of a graph is just a subgraph that contains all the vertices and is a tree. A minimum spanning tree is a spanning tree with weight less than or equal to the weight of every other spanning tree.

The principle is based on the Prim's algorithm:

```
Select an arbitrary node to start
While (there are fringe vertices)
     select minimum-weight edge between tree and fringe
     add the selected edge and node to the tree
```

Edges between nodes are physically modified: the neighbor relationships are lost and the weight of the remain edges are set to 1.

Inputs

- *gr_in*: a graph.

Outputs

- *gr_out*: a graph.

Result

Returns SUCCESS or FAILURE.

Examples

```
pmst g1.pan g2.pan
```

See also

Graph

C++ prototype

```
Errc PMst( const Graph2d &gr_in, Graph2d &gr_out );
```

Version française

Construction de l’arbre de recouvrement minimal d’un graphe.

Author: François Angot

pmult

Performs multiplication between images or graphs.

Synopsis

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

Description

pmult performs the multiplication of the two inputs *im_in1* and *im_in2*.

If *im_in1* and *im_in2* are images then the new image *im_out* is built with the multiplication of each pixel:

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

The two inputs must be of the same type.

The output type *im_out* depends on input type:

- Long if inputs are Uchar images.
- Long if inputs are Long images.
- Float if inputs are Float images.

For color or multispectral image, the multiplication is computed separately on each band.

If *im_in1* and *im_in2* are graphs then the new graph *im_out* is built with the multiplication of each node values.

Inputs

- *im_in1*: an image or a graph.
- *im_in2*: an image or a graph.

Outputs

- *im_out*: an image or a graph.

Result

Returns SUCCESS or FAILURE.

Examples

```
pmult a.pan b.pan c.pan
```

See also

Arithmetic

C++ prototype

```
Errc PMult( const Img2duc &im_in1, const Img2duc &im_in2, Img2dsf
&im_out );
```

Version française

Multiplication d’images ou de graphes.

Author: Régis Clouard

pmultcst

Multiplies constant to image, graph or region map.

Synopsis

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

Description

pmultcst builds the new output *im_out* by multiplying the specified constant to each value of *im_in*.

For image, **pmultcst** multiplies the specified value to each pixel. The values are clipped if they are greater than the maximum possible value or lower than the minimum:

```
val = pixel(im_in) * cst;
if (val > MAX) pixel(im_out) = MAX;
else if (val < MIN) pixel(im_out) = MIN;
else pixel(im_out) = val;
```

For color or multispectral image, **pmultcst** is computed separately on each band.

For region map, **pmultcst** multiplies the specified value to each label.

For graph, **pmultcst** multiplies the specified value to each node value.

The output file is of the same type as the input file.

Parameters

- *cst* is a real value.

Inputs

- *im_in*: an image, a graph or a region map.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

For region map, returns the new higher label value.

Examples

Divides the tangram.pan pixel values by 2:

```
pmultcst 0.5 tangram.pan a.pan
```

See also

Arithmetic

C++ prototype

```
Errc PMultCst( const Img2duc &im_in, Img2duc &im_out, float cst );
```

Version française

Multiplication par une constante des valeurs d’une image, d’un graphe ou d’une carte de région.

Author: Régis Clouard

pmultval

Multiplies image bands with constants stored in collection.

Synopsis

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

Description

pmultval builds the new output *im_out* by multipling each band of the input image *im_in* with constants stored in the collection *col_in*. The first bands is multiplied with the first constant in the collection, the second bdans with the second constant, etc.

The values are clipped if they are greater than the maximum allowed value or lower than the minimum:

```
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;
```

The output file is of the same type as the input file.

Inputs

- *col_in*: a collection with a number of float values equals to the number of bands of the input image.
- *im_in*: an image.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Multiplies tangram.pan by its mean value:

```
pmeanvalue tangram.pan col.pan
pmultval col.pan tangram.pan a.pan
```

More examples

See also

Arithmetic

C++ prototype

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

Version française

Multiplication d’une image par des constantes stockÃ©es dans une collection.

Author: Régis Clouard

pmumfordshahmerging

Performs priority region merging based on Mumford-Shah criterion.

Synopsis

pmumfordshahmerging *number alpha threshold* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

pmumfordshahmerging merges connected regions of the input image *rg_in* if the difference between the energy variation of the region defined by Mumford and Shah is lower than the specified *threshold*.

Two regions are connected if there exists a link between the related nodes in the input graph *gr_in*.

The principle of the algorithm is as follows:

- For each region of the input region map *rg_in*:
- If the difference between the criterion value of the two connected regions ≤ *threshold* then merge them into one region.

The algorithm uses the priority merging that consists in merging regions with the lower difference.

The output region map *reg_out* defines the new regions and the output graph *gr_out* defines the new relationship between regions.

The energy variation (DE) is calculated as follows :

$$dE = \frac{Card(R1)*Card(R2)*(mean(R1)-mean(R2))^2 - 2*alpha*boundary(R1,R2)}{Card(R1)+Card(R2)}$$

where *alpha* is a parameter;
mean(R1) is the mean of the region R1;
boundary is the length of the boundary between R1 and R2;
and Card(R1) is the number of pixels in the region R1.

Negative values mean that energy that results from the merging of R1 and R2 is lower than the sum of the energy of the two regions.

Parameters

- *number* specifies the number of allowed merging. If *number* = -1 then all possible merging are done.

- *alpha* specifies the importance given to the boundary of the regions rather than the inner mean difference. Values are positive values and can be greater than 2000.
- *threshold* specifies the maximum energy difference allowed between two regions to decide to merge them. Values can be positive or negative values. A typical value is 0.

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.
- *gr_out*: a graph.

Result

Returns the number of merging.

Examples

Merges regions yielded by a quadtree splitting process:

```
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
```

See also

Segmentation

C++ prototype

```
Errc PMumfordshahMerging( const Reg2d &rg_in, const Graph2d &gr_in,
const Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, long number,
double alpha, float threshold );
```

Version française

Fusion prioritaire de régions selon la variation d’énergie de Mumford Shah.

Author: Laurent Quesnel

pnagaofiltering

Performs Nagao filtering on image.

Synopsis

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

Description

pnagaofiltering applies the Nagao filter algorithm on the input image *im_in*.

The Nagoa filter proceeds by partitioning the neighborhood into distinct domains from which an homogeneous criterion is calculated. The more homogeneous domain is chosen and then central pixel is replaced by the mean value of the domain. Nagao filter uses 9 domains of size 5x5: the basis matrix and its 8 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.
```

The image border (of size halfsize) is not considered for processing. The output image border is just a copy of the input image border.

Inputs

- *im_in*: a grayscale 2d image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Applies a Nagao filter to tangram.pan:

```
pnagaofiltering tangram.pan out.pan
```

See also

Filtering

C++ prototype

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

Version française

Lissage par maximum d’homogénéité selon le masque de Nagao.

Author: Régis Clouard

pnewimage

Creates a new image from dimensions.

Synopsis

pnewimage *w h d val* [*im_out*|-]

Description

pnewimage creates a new image from the specified dimensions (*width*,*height*,*depth*) with the specified value *val* for all pixels.

The format of the output image is deduced from the dimension values:

- if *h*<=0 then *im_out* is a 1D image;
- if *d*<=0 then *im_out* is a 2D image;
- otherwise *im_out* is a 3D image.

The type of the output image is deduced from the parameter *val* value:

- if *val* is an integer < 255 and ≥ 0 then *im_out* is a Uchar image;
- if *val* is an integer > 255 or < 0 then *im_out* is a Long image;
- if *val* is a real value then *im_out* is a Float image.

Two other operators can be used to create a new image. psetcst creates a new image from properties of an another image. pshapedesign creates a new image from specified dimensions and type.

Parameters

- *w*,*h*,*d* (width, height, depth) defines the dimensions of the output image.
- *val* specifies the value for all pixels. It can be an integer or a real value.

Outputs

- *im_out*: an image whose type depends on the parameters values.

Result

Returns SUCCESS or FAILURE.

Examples

Fills hole in regions yields by a simple segmentation process of the 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
```

See also

Utility, pshapedesign, psetcst

C++ prototype

```
No prototype.
```

Version française

Création d’une nouvelle image à partir de dimensions.

Author: Régis Clouard

pnodedisc

Visualizes graph node values.

Synopsis

pnodedisc [*gr_in*|-] [*rg_out*|-]

Description

pnodedisc allows the visualization of the value associated to each node of the input graph *gr_in*. Each node value is visualized by the way of a disc with a radius proportional to its value. The label of the disc is the rank of the node in the list of nodes.

Inputs

- *gr_in*: a graph.

Outputs

- *rg_out*: a region map.

Result

Returns the number of regions.

Examples

```
pnodedisc g1.pan r.pan
```

See also

Graph

C++ prototype

```
Errc PNodeDisc( const Graph &gr_in, Reg2d &rg_out );
```

Version française

Visualisation des valeurs des noeuds d’un graphe.

Author: François Angot

pnodevisu

Visualizes graph node values.

Synopsis

pnodevisu [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_out*|-]

Description

pnodevisu allows the visualization of the value associated to each node of the input graph *gr_in*. The output image is built from the region of the input region map *rg_in* where each region is colored with the value of the related node in the input graph *gr_in*.

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.

Outputs

- *im_out* : an image.

Result

Returns SUCCESS or FAILURE.

Examples

```
pnodevisu r.pan g.pan i.pan
```

See also

Graph

C++ prototype

```
Errc PNodeVisu( const Reg2d &rg_in, const Graph2d &gr_in, Img2dsl
&im_out );
```

Version française

Visualisation des valeurs des sommets d’un graphe dans une image.

Author: François Angot

pnonmaximasuppression

Performs non-maxima suppression for edge detection.

Synopsis

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

Description

pnonmaximasuppression finds the non maxima values from the magnitude gradient image *im_in1*. Maxima values are gradient magnitude that are maximum in the orthogonal direction of the gradient.

The direction image *im_in2* contains pixels that are set with Freeman codes.
The Freeman codes are:

```
   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
```

Note: the output border values (size 1x1x1) are set the input image border values. *im_out*.

The output image is of the same type as the input image.

Inputs

- *im_in1*: a grayscale image.
- *im_in2*: a byte image (Uchar image).

Outputs

- *im_out*: an image of the same type as *im_in1*.

Result

Returns SUCCESS or FAILURE.

Examples

Extracts contours from the tangram.pan image:

```
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
pgeodesicdilatation 1 1 -1 i6.pan i7.pan out.pan
```

See also

Edge detection

C++ prototype

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

Version française

Suppression des points non maxima dans une image d’amplitude de gradient.

Author: Régis Clouard

pnormalization

Performs normalization of image or graph.

Synopsis

pnormalization *min max* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pnormalization computes the normalization image's values or the graph's value between new values *min* et *max*.

If *im_in* is an image, pnormalization is applied on each pixel value:

```
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))];
```

For color or multispectral image, the normalization is computed separately on each band.

The output image *im_out* is of the same type as *im_in*.

If *im_in* is a graph then the new graph *im_out* is built with the normalization of each node value.

Parameters

- *min* and *max* are real values. Their values must be in the value domain defined by the input object type (eg., 0..255 for Uchar image, ...)

Inputs

- *im_in*: an image or a graph.

Outputs

- *im_out*: an image or a graph of the same type as *im_in.*

Result

Returns SUCCESS or FAILURE.

Examples

Normalizes tangram.pan pixel between 10 and 234:

```
pnormalization 10 234 tangram.pan a.pan
```

See also

Arithmetic

C++ prototype

```
Errc PNormalization( const Img2duc &im_in, Img2duc &im_out, Uchar
min, Uchar max );
```

Version française

Normalisation d’une image entre deux valeurs extrèmes.

Author: Régis Clouard

pnot

Performs logical negation of image or graph and complementary of region map.

Synopsis

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

Description

pnot performs the logical negation of the input.

If input is an image, the negation operator uses the '!' C operator. If an input pixel is > 0 then output pixel is 0. If an input pixel is 0 then the output pixel is 1.

```
pixel(im_out) = ! pixel(im_in)
```

For color or multispectral image, the "not" operator is computed separately on each band.

For the graph, the negation operator ! is applied to each node values.

For region map, **pnot** performs the complementary of the region map *im_in*. Regions with labels > 0 in *im_in* become region with label =0 in the *im_out* and region with label =0 becomes region with label=1. Result regions are not necessarily composed of connected components. There are defined by components with the same label.

Inputs

- *im_in*: an image, a graph or a region map.

Outputs

- *im_out*: an object of the same type as im_in.

Result

Returns SUCCESS or FAILURE.

For region map, returns the new higher label value (0 or 1).

Examples

```
pnot a.pan b.pan
```

See also

logic

C++ prototype

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

Version française

Négation logique d’image ou de graphe et complémentaire d’une carte de régions.

Author: Régis Clouard

pobject2col

Creates collection from Pandore object.

Synopsis

pobject2col *name* [*obj_in*|-] [*col_out*|-]

Description

pobject2col creates a new collection that contains the specified Pandore object *obj_in* referenced by the specified *name*.

Parameters

- *name* is the name of Pandore object in the collection. Is a string without blank character.

Inputs

- *obj_in*: a Pandore file.

Outputs

- *im_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Adds tangram.pan image to collection col.pan:

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

See also

Collection, pcolsetobject

C++ prototype

```
Errc PObject2Col( const Img2duc &obj_in, Collection &col_out, const
std::string &name );
```

Version française

Création d’une collection contenant un objet Pandore.

Author: Alexandre Duret-Lutz

popencontourselection

Selects open contour from length.

Synopsis

popencontourselection *relation length* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

popencontourselection selects open contours from their *length*. An open contour is a chain of connected non null pixels with 1 pixel thickness that begins and ends with an end point (a point with only one neighbor) and which has no junction. A closed contour and a barb are not considered as open contour.

The parameter *relation* specifies the relation order to the *length* value that is used to select or not a contour.

Warning: if the contour is not 1 pixel thickness, the operator may have unpredictable behavior. It might be necessary to use the operator ppostthinning to guaranty 1 pixel thickness.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *length* value:
  - *relation* = 3: contours with the maximum length.
  - *relation* = 2: contours >= *length*.
  - *relation* = 1: contours > *length*.
  - *relation* = 0: contours = *length*.
  - *relation* = -1: contours < *length*.
  - *relation* = -2: contours <= *length*.
  - *relation* = -3: contours with the minimum length.
- *length* is an integer defined in pixel unit.

Inputs

- *im_in*: a grayscale image (Img2duc or Img3duc) which contains the contours.

Outputs

- *im_in*: a grayscale image (Img2duc or Img3duc).

Result

Returns the number of selected open contours.

Examples

Selects open contours from contours yielded by a simple edge detection of tangram.pan:

```
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
```

See also

Contour

C++ prototype

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

Version française

Selection des chaînes de contours ouvertes sur leur longueur.

Author: Régis Clouard

por

Performs binary or between images or graphs and union between region maps.

Synopsis

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

Description

por performs a bitwise "or" between values of the two inputs *im_in1* and *im_in2*.

If inputs are integer images, the "or" operator uses the ’|’ C operator and is applied on each pixel:

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

For real images, the "or" operator is "+":

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

For color or multispectral image, the "or" operator is computed separately on each band.

For graph, the "or" operator is "+" and it is applied on each node values.

For region map, the "or" operator is the union between regions. The result *im_out* is a new region map with regions of each input region maps, given preference to smaller regions.

The two inputs must be of the same type.

Inputs

- *im_in1*: an image, a graph or a region map.
- *im_in2*: an image, a graph or a region map.

Outputs

- *im_out*: an object of the same type as im_in1 and im_in2.

Result

Returns SUCCESS or FAILURE.

For region map, returns the new higher label value.

Examples

```
por a.pan b.pan c.pan
```

See also

logic

C++ prototype

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

Version française

Ou binaire entre images ou graphes, et union entre cartes de régions.

Author: Régis Clouard

porientationselection

Selects regions from orientation degree.

Synopsis

porientationselection *relation threshold* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

porientationselection selects regions from their orientation. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

The orientation value is specified in degree. Is is computed from the moments:

```
orientation = 0.5*arctan(2*M11/(M20-M02)).
```

If `M20=M02` the region is symmetrical and orientation is every degree.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is an integer from [0..360] defined in degree.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *rg_out*: a 2D region map.

Result

Returns the number of selected regions.

Examples

Selects vertical regions from the region map a.pan:

```
porientationselection 90 a.pan b.pan
```

See also

Region

C++ prototype

```
Errc POrientationSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, Ushort threshold );
```

Version française

Sélection de régions sur leur valeur d’orientation.

Author: Régis Clouard

poutborderselection

Selects regions that do not touch the border.

Synopsis

poutborderselection *w h d* [*rg_in*|-] [*rg_out*|-]

Description

poutborderselection builds the new region map *rg_out* from the region of the input region map *rg_in* that do not touch the border of the region map. The border is specified from the depth *p*, the height *h* and the width *l* dimensions.

Regions are not relabeled, they keep the same label as in the input region map.

Parameters

- *w, h d* specify respectively the width, the height and the depth of the border.

Inputs

- *rg_in*: a region map.

Outputs

- *rg_out*: a region map.

Result

Returns the number of selected regions.

Examples

Discards regions that touch the border of the 2D region map:

```
poutborderselection 1 1 0 rin.pan rout.pan
```

See also

Region

C++ prototype

```
Errc POutBorderSelection( const Reg2d &rg_in, Reg2d &rg_out, int d,
int h, int w );
```

Version française

Sélection des régions qui ne touchent pas le bord de l’image.

Author: Régis Clouard

poutrangefiltering

Performs outrange filtering on image.

Synopsis

poutrangefiltering *eps* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

poutrangefiltering applies the outrange filter algorithm on the input image *im_in*.

Each pixel is replaced by the mean value of its neighbors if the difference between this mean and the central pixel value is lower or equal than the specified parameter value *eps*. If *eps*=255 than outrange is equivalent to meanfiltering.

The image border (of size 1) is not considered for processing. The output image border is just a copy of the input image border.

Parameters

- *eps* specifies the maximum difference allowed from the central pixel and the mean value of its neighbors. It is a grayscale value.

Inputs

- *im_in*: a grayscale image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Applies a sigma filter to tangram.pan:

```
poutrangefiltering 10 tangram.pan out.pan
```

See also

Filtering

C++ prototype

```
Errc POutRangeFiltering( const Img2duc &im_in, Img2duc &im_out,
float eps );
```

Version française

Lissage par filtre adaptatif basé sur le choix des voisins.

Author: Régis Clouard

ppan2analyze

Converts Pandore image file to ANALYZE 7.5 image file.

Synopsis

ppan2analyze *im_in* [*im_out*|-]

Description

ppan2analyze generates an ANALYZE 7.5 image from a Pandore image.

An ANALYZE 7.5 image is composed of 2 files in the same directory and with the same base name:

- header file (suffixed .hdr)
- data file (suffixed .img).

im_out is just the base name of the ANALYZE file, it means without suffix.

Inputs

- *im_in*: a Pandore image.

outputs

- *im_out*: the base name of an ANALYZE 7.5 file.

Result

Returns SUCCESS or FAILURE.

Examples

Converts the Pandore image a.pan to a ANALYZE image "brain.hdr" and "brain.img":

```
pan2analyze a.pan brain
```

See also

Conversion, panalyze2pan

C++ prototype

```
Errc PPan2Analyze( const Imx3d &img, const char *filename_out );
```

Version française

Conversion d’une image Pandore en image ANALYZE 7.5.

Important notice

Author: David Tschumperlé

ppan2bmp

Converts Pandore file to BMP image file.

Synopsis

ppan2bmp [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

ppan2bmp converts a Pandore image file to a BMP (Bitmap) image file.

Only the following Pandore image types can be converted:

- 2D gray level image of bytes (Img2duc);
- 2D color image of bytes (Imc2duc);
- 2D region map (Reg2d).

Other Pandore image types might be converted to regular types using convenient casting operators.

Inputs

- *im_in*: a 2D image (img2duc, Imc2duc) or a 2D region map.

Outputs

- *im_out*: a BMP image file.

Result

Returns SUCCESS or FAILURE.

Examples

```
ppan2bmp tangram.pan image.bmp
```

See also

Conversion, pbmp2pan

C++ prototype

```
Err PPan2Bmp( const Img2duc &im_in, char *filename );
```

Version française

Conversion d’une image 2D Pandore en image BMP.

Author: Régis Clouard

ppan2d23d

Converts a series of Pandore 2D image files to a unique 3D image file.

Synopsis

ppan2d23d *begin end im_in* [*im_out*|-]

Description

ppan2d23d builds a 3D image from a series of 2D images. Each 2D image becomes a plane in the 3D image.

The template name *im_in* is used to specify actual 2D image files. Special character '#' in the template name are replaced by a number from the interval [*begin* .. *end*]. For example:

- `toto####.pan` with begin=8 and end=10 specify files: `toto0008.pan, toto0009.pan, toto0010.pan`.
- `toto#.pan` with begin=8 and end=10 specify files: `toto8.pan, toto9.pan, toto10.pan`.

Parameters

- *begin* et *end* specify respectively the minimum number and the maximum number of real 2D images. There is no relation between the number of the input image and the plane number in the 3D image. The first image becomes the plane #0 and the last image becomes the last plane.

Inputs

- *im_in*: a template name for 2D Pandore image files. It uses the special character # for specifying number format in the name.

Outputs

- *im_out*: a 3D Pandore image file.

Result

Returns SUCCESS or FAILURE.

Examples

```
ppan2d23d 0 10 image##.pan image3d.pan
```

See also

Conversion, ppan3d22d

C++ prototype

```
Errc PPan2d23d( const char *2dname, const char *3dname, int begin,
int end );
```

Version française

Conversion d’une série d’images 2D en une image 3D.

Author: François Angot

ppan2fits

Converts Pandore gray-scale image (1D, 2D or 3D) to FITS (Flexible Image Transport System) image file.

Synopsis

ppan2fits [*im_in*|-] [*im_out*|-]

Description

ppan2fits converts a gray-scale Pandore image to the FITS (Flexible Image Transport System) image file.

Only 1D, 2D and 3D gray-scale images are considered at the moment.

Input

- *im_in*: a Pandore image file.

Output

- *im_out*: a FITS image file.

Result

Returns SUCCESS or FAILURE.

Examples

Converts the andore image to FITS image:

```
ppan2fits tangram.pan tangram.fits
```

See also

Conversion, pfits2pan

C++ prototype

```
Errc PPan2Fits( const Img3dsf &ims, char *filename );
```

Version française

Conversion d’une image Pandore vers le format FITS.

Author: Jalal Fadili

ppan2gif

Converts Pandore image file to GIF image file.

Synopsis

ppan2gif [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

ppan2gif converts 2D Pandore image file to GIF image file.

Only the following Pandore image types can be converted:

- 2D gray level image of bytes (Img2uc);
- 2D color image of bytes (Imc2duc);
- 2D region map (Reg2d).

Other Pandore image types might be converted in regular types using convenient casting operators.

Inputs

- *im_in*: a 2D Pandore image file (img2duc, Imc2duc) or a 2D region map.

Outputs

- *im_out*: a GIF image file.

Result

Returns SUCCESS or FAILURE.

Examples

```
ppan2gif tangram.pan image.gif
```

See also

Conversion, pgif2pan

C++ prototype

```
Errc PPan2Gif( const Img2duc &im_in, char *filename );
```

Version française

Conversion d’une image pandore 2D en image GIF.

Author: Régis Clouard

ppan2pan

Converts any Pandore file format to current Pandore file format.

Synopsis

ppan2pan [*im_in*|-] [*im_out*|-]

Description

ppan2pan converts an old Pandore file format to the current Pandore file format. Old Pandore file format are Pandore2 and Pandore3 formats.

Old Pandore files format are not considered with the current operators. Therefore it is necessary to convert such files.

Inputs

- *im_in*: a Pandore image file.

Outputs

- *im_out*: a Pandore image file.

Result

Returns SUCCESS or FAILURE.

Examples

Converts an old Pandore image format to current Pandore image format:

```
ppan2pan oldtangram.pan tangram.pan
```

See also

Conversion

C++ prototype

```
Errc PPan2pan( const FILE* fdin, Pobject **objout );
```

Version française

Conversion d’un fichier Pandore en un fichier Pandore.

Author: Régis Clouard

ppan2ppm

Converts Pandore image file to PPM (ascii) image file.

Synopsis

ppan2ppm [*im_in*|-] [*im_out*|-]

Description

ppan2ppm converts the Pandore file *im_in* to the PPM file format *im_out*.

A PPM consists of the followings:

- The first line contains the "magic number" among:
  - P2: for gray scale image with ASCII data.
  - P5: for gray scale image with pure binary data.
  - P3: for color image with ASCII data.
  - P6: for color image with pure binary data.
- The second line contains the number of columns and rows in ASCII.
- The third line contains the number of colors.

  Each line can include comments introduced by #.

- The data format depends on the magic number: (Caution: no line should be longer than 70 characters.)
  - P2: pixel is represented as an ASCII decimal number.
  - P5: pixel is represented as 1 byte.
  - P3: pixel is represented as a triplet of ASCII decimal number (Red, Green Blue).
  - P6: pixel is represented as a triplet of bytes (Red, Green Blue).

Example:

```
P2
#Gray scale image
#with 5 columns and 5 rows.
5 4
255
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

Inputs

- *im_in*: a Pandore image file.

Outputs

- *im_out*: a PPM (color) image file or PGM (gray level) image file.

Result

Returns SUCCESS or FAILURE.

Examples

```
ppan2ppm tangram.pan tangram.ppm
```

See also

Conversion, pppm2pan

C++ prototype

```
Errc PPan2PPM( const Img2duc &ims, char* f_out );
```

Version française

Conversion d’un fichier Pandore en un fichier de format PPM (Portable PixMap ASCII) ou PGM (Portable GrayMap ASCII).

Author: Régis Clouard

ppan2ps

Converts Pandore image file to Encapsulted PostScript file.

Synopsis

ppan2ps [*im_in*|-] [*file_out*|-]

Description

ppan2ps converts Pandore image file to PostScript file. Only 2D images can be converted to PostScript files.

The result file can then be printed or included in TeX file.

Inputs

- *im_in*: a 2D image file or a 2D region map.

Outputs

- *file_out*: a PostScript file.

Result

Returns SUCCESS or FAILURE.

Examples

Builds a PostScript file with the Pandore image tangram.pan:

```
ppan2ps tangram.pan tangram.eps
```

See also

Conversion

C++ prototype

```
Errc PPan2PS( const Img2duc &im_in, char *filname );
```

Version française

Conversion d’une image Pandore en un fichier Encapsuled PostScript.

Author: Régis Clouard

ppan2raw

Converts Pandore file to raw file.

Synopsis

ppan2raw [*im_in*|-] [*file_out*|-]

Description

ppan2raw converts Pandore image to binary image.

Raw file are composed of pure binary data:

- For 2D images, pixels are ordered by line and for each line by columns.
- For 3D images, pixels are ordered by planes, lines and columns.
- For color images, data are organized by color: first red pixels, then green pixels and finally blue pixels.
- For multispectral images, data are organized by bands: first band #0, then band #1, etc.

Inputs

- *im_in*: a Pandore image file.

Outputs

- *file_out*: a binary image file.

Result

Returns SUCCESS or FAILURE.

Examples

```
ppan2raw tangram.pan tangram.raw
```

See also

Conversion, praw2pan

C++ prototype

```
Errc PRaw2Pan( const Img2duc &im_in, char *filename );
```

Version française

Conversion d’un fichier image Pandore en un fichier image sans entête.

Author: Régis Clouard

ppan2tiff

Converts Pandore image file to TIFF image file.

Synopsis

ppan2tiff [*-m mask*] [*im_in*|-] *im_out*|-]

Description

ppan2tiff converts a Pandore file to TIFF image file. The conversion can generate 8 or 16 bits TIFF image that corresponds to Uchar and Long Pandore images either in gray scale or color.

In case of Long images (Img2dslm, Img3dsl, Imc2dsl or Imc3dsl), pixels are clipped to convert pixel with 4 bytes to pixel with 2 bytes (0..65535): pixel with value > 65535 is set to 65535 and pixel with value < 0 is set to 0.

In case of 3D images, **ppan2tiff** builds as many files s they are planes. Files are named: "im_out*ndep*.tiff" where *ndep* is the number of plane of the 3D image. For example **ppan2tiff** a.pan img.tiff, yields files: `img000.tiff, img001.tiff ...`

Inputs

- *im_in*: a Pandore image file or a Pandore region map file.

Outputs

- *im_out*: a TIFF image file.

Result

Returns SUCCESS or FAILURE.

Examples

```
ppan2tiff tangram.pan image.tiff
```

See also

Conversion, ptiff2pan

C++ prototype

```
Errc PPan2Tiff( const Img2duc &im_in, char *im_out );
```

Version française

Conversion d’une image Pandore en image(s) TIFF.

Author: Régis Clouard

ppan2txt

Converts Pandore image file to text file.

Synopsis

ppan2txt [*-m mask*] [*im_in*|-] [*file_out*|-]

Description

ppan2txt builds a text file with all non null pixels of the input Pandore file *im_in*. The result text file is organized as follows for a 3D image

```
value x y z
```

or as follows for a 2D image:

```
value x y
```

Inputs

- *im_in*: a gray scale image (1D, 2D, or 3D).

Outputs

- *file_out*: a text file.

Result

Returns SUCCESS or FAILURE.

Examples

Builds a text file from the Pandore image tangram.pan:

```
ppan2txt tangram.pan image.txt
```

See also

Conversion, ptxt2pan

C++ prototype

```
Errc PPan2txt( const Img3duc &im_out, char *nom, Long z, Long y Long
x );
```

Version française

Conversion d’une image en une liste de points dans un fichier texte.

Author: Régis Clouard

ppan2vff

Converts Pandore image file to VFF image file (Sunvision).

Synopsis

ppan2vff [*-m mask*] [*im_in*|-] [*file_out*|-]

Description

ppan2vff converts Pandore file to VFF (Sunvision) image file.

A VFF image is a 3D image of bytes.

For Long and Float images, **ppan2vff** clips the value between [0..255]. Il might be necessary to cast the value beforehand using convenient casting Pandore operators.

Inputs

- *im_in*: a 3D Pandore image file or a 3D Pandore region map file.

Outputs

- *file_out*: a VFF image file.

Result

Returns SUCCESS or FAILURE.

Examples

Converts the Pandore 3D image cube.pan to VFF image file:

```
ppan2vff cube.pan image.vff
```

See also

Conversion, pvff2pan

C++ prototype

```
Errc PPan2Vff( const Img3duc &im_in, char *filename );
```

Version française

Conversion d’une image Pandore 3D en une image au format VFF.

Author: François Angot

ppan3d22d

Converts Pandore 3D image file to a series of 2D Pandore image files.

Synopsis

ppan3d22d [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

ppan3d22d extracts each plane of a 3D image to build a series of 2D images.

The name of the 2D image files are composed from the base name *im_out* + the plane number + the suffix of *im_out*. The plane number of plane is padded with leading 0 so as to force numbers to occupy the same size.

For example, the command `pan3d22d a.pan b.pan` yields images:

- if `a.pan` contains 13 slices : `b00.pan, b01.pan ... b12.pan`
- if `a.pan` contains 121 slices : `b000.pan, b001.pan ... b120.pan`

Inputs

- *im_in*: a 3D Pandore image file.

Outputs

- *im_out*: the template name of 2D Pandore image files.

Result

Returns SUCCESS or FAILURE.

Examples

```
ppan3d22d cube.pan image.pan
```

See also

Conversion, ppan2d23d

C++ prototype

```
Errc PPan3d22d( const Img3duc &im_in, char *filename );
```

Version française

Conversion d’une image Pandore 3D en une série d’images Pandore 2D.

Author: François Angot

pparrec2pan

Converts PAR/REC format image file (Philips Medical System) to Pandore image file.

Synopsis

pparrec2pan *im_in* [*im_out*|-]

Description

pparrec2pan converts a PAR/REC (Philips Medical System) image file to a Pandore image file.

A PAR/REC image is composed of 2 files:

- a header file (suffixed .par)
- a data file (suffixed .rec).

The input file *im_in* is one of the two PAR/REC files. The second file is then read from the same directory and with the same base name.

The result image *im_out* is always a 3D multispectral image of floats (Imx3dsf).

Inputs

- *im_in*: a PAR/REC file (either .par or .rec file)

Outputs

- *im_out*: a Pandore image file (Imx3dsf).

Result

Return SUCCESS or FAILURE.

Examples

Converts image "brain" to Pandore image "a.pan" and then displays the band #0.

```
parrec2pan brain.par a.pan
pimx2img 0 a.pan out.pan
```

See also

Conversion

C++ prototype

```
Errc PParrec2Pan( const char *filename, Pobject **obj_out );
```

Version française

Conversion d’une image au format PAR/REC (Philips Medical System) en image Pandore.

Important notice

Author: David Tschumperlé

ppeergroupfiltering

Performs Peer Group filtering on color image.

Synopsis

ppeergroupfiltering *threshold* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

ppeergroupfiltering applies a Peer Group filter to the input image *im_in*. Each pixel is replaced by the weighted average of its peer group members which are classified based on the color similarity of its neighboring pixels.

Parameters

- *threshold* specifies the magnitude of the noise to be removed. It is a positive real value. A typical value is 1.0.

Inputs

- *im_in*: a color image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Applies the Peer Group Filtering filter on parrot.pan image:

```
ppeergroupfiltering 10 parrot.pan out.pan
```

See also

Filtering

C++ prototype

```
Errc PPeerGroupFiltering( const Imc2duc &im_in, Imc2duc &im_out,
float threshold );
```

Version française

Lissage d’une image couleur par Peer Group Filtering.

Author: Olivier Lezoray

pperimeterselection

Selects regions from perimeter length.

Synopsis

pperimeterselection *relation threshold* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

pperimeterselection selects regions from their perimeter length. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

The region perimeter is the number of pixels on the region boundary. The algorithm uses one quarter pixel for concavity. For example, the perimeter is 7 pixels for the following region (`7=6+4*0.25`) :

```
 xx
xxxx
 xx
```

For a region with 1 pixel, the perimeter is 1.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is a positive integer defined in pixel unit.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *rg_out*: a 2D region map.

Result

Returns the number of selected regions.

Examples

Selects regions that have a boundary > 50 pixels:

```
pperimeterselection 1 50 rin.pan rout.pan
```

See also

Region

C++ prototype

```
Errc PPerimeterSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, Ulong threshold );
```

Version française

Sélection de régions sur leur valeur de périmètre.

Author: Régis Clouard

pphase

Computes phase between 2 images.

Synopsis

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

Description

pphase computes the phase between 2 images considered as a unique complex image.

The phase between 2 images is calculated as follows:

```
pixel(im_out) = atan(pixel(im_in2) / pixel(im_in1))
```

Inputs

- *im_in1*: a grayscale or a color image.
- *im_in2*: a grayscale or a color image.

Outputs

- *im_out*: a float image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts an image of a square from the spatial domain to the frequency domain and reciprocally:

```
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
```

See also

Frequency, pmodulus

C++ Prototype

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

Version française

Calcul de la phase entre deux images.

Author: Régis Clouard

ppixelvalue

Gets pixel value from grayscale image and region map.

Synopsis

ppixelvalue *x y z* [*-m mask*] [*im_in*|-] [*col_out*|-]

Description

ppixelvalue returns the value at the specified coordinates (*x*,*y*,*z*) in the input image or in the input region map *im_in*.

This value can be get using operator **pstatus**.

The pixel values for each band are stored in the collection *col_out*.

Parameters

- *x*, *y*, *z* specify the coordinate of the pixel. For 2D image, *z* is ignored but must be given.

Inputs

- *im_in*: an image or a region map.

Outputs

- *col_out*: a collection of float values.

Result

Returns the pixel value or FAILURE if the coordinates is not held by the input image (for the first band only). This value can be get using operator **pstatus**.

Examples

Returns the value of the pixel 20,50 in tangram.pan (Unix version):

```
ppixelvalue 20 50 0 tangram.pan col.pan
val=‘pstatus‘
echo "value = $val"
```

Returns the value of the pixel 20,50 in tangram.pan (MsDOS version):

```
ppixelvalue 20 50 0 tangram.pan col.pan
call pstatus
call pset val
echo Value = %val%
```

See also

Image Features Extraction

C++ prototype

```
Float PPixelValue( const Img3duc &im_in, Collection & col_out, Long z, Long y, Long x );
```

Version française

Affiche la valeur d’un pixel d’un image ou d’un noeud d’un graphe donné.

Author: Régis Clouard

pplot1d

Renders a plot of a 1D function as a 2D color image.

Synopsis

pplot1d *dimx dimy type ymin ymax* [*im_in*|-] [*im_out*|-]

Description

pplot1d draws the representation of a 1D function as a 2D image. Parameters *dimx* and *dimy* specify the output image dimension.

The representation can take several forms according to the parameter *type*.

Parameters

- *dimx* specifies the width the 2D representation.
- *dimy* specifies the height the 2D representation.
- *type* specifies the type of representation among:
  - 0 = lines (piecewise continuous).
  - 1 = bar diagram.
  - 2 = lines (bicubic interpolation).
- *ymin* specifies the minimum value of the y axis.
- *ymax* specifies the maximum value of the y axis. **Note**; If ymin=ymax=0 then the scale is automatically calculated from the input data values.

Inputs

- *im_in*: a grayscale 1D image.

Outputs

- *im_out*: a color 2D image (Imc2duc)

Result

Returns SUCCESS or FAILURE.

Examples

Displays the histogram of the tangram.pan image as a 1D function:

```
phistogram tangram.pan a.pan
pplot1d 512 256 0 0 0 a.pan b.pan
pvisu b.pan
```

See also

Visualization

C++ prototype

```
Errc PPlot1d( const Img1d &ims, Imc2duc &imd, Uchar type,Float ymin,
Float ymax );
```

Version française

Construction d’une image 2D à partir d’une fonction 1D.

Inportant notice

Author: D. Tschumperlé

pplotquiver

Renders vector field from multispectral images with 2 bands.

Synopsis

pplotquiver *dimx dimy sampling factor* [*im_in*|-] [*im_out*|-]

Description

ppplotquiver draws a vector field in the 2D image *im_out* from the vector components given in the multispectral image *im_in*. The first band of *im_in* contains the x-component of the vector and the second band contains the y-component. Such an image can be built with the operator pregistrationPDE.

Parameters

- *dimx* defines the width of the output image.
- *dimy* defines the height of the output image.
- *sampling* defines the gap (in pixel) between 2 consecutive vectors.
- *factor* is a multiplicative factor applied to vectors length. If the value is negative then it corresponds to a percentage of the maximum vector length.

Inputs

- *im_in*: a multispectral 2D image with at least 2 bands. If the number of bands is greater than 2 then other bands are ignored.

Outputs

- *im_out*: an image (Img2duc).

Result

Returns SUCCESS or FAILURE.

Examples

Displays the vector field of a translation of the tangram.pan image:

```
ptranslation 0 10 tangram.pan tangram1.pan
pregistrationPDE 0.1 0.9 tangram.pan tangram1.pan a.pan
pplotquiver 800 800 10 -20 a.pan out.pan
```

See also

Motion

C++ prototype

```
Errc PPlotQuiver( const Imx2d &ims, Img2duc &imd, Short sampling,
Float factor );
```

Version française

Dessine un champ de vecteurs 2D à partir d’une image 2D multi-spectrale à deux composantes.

Important notice

Author: D. Tschumperlé

ppolygonalapproximation

Performs polygonal approximation of contours.

Synopsis

ppolygonalapproximation *error* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

ppolygonalapproximation approximates a given set of piecewise linear curves as 2D polygons. A curve is a chain of non null pixels and of 1 pixel thickness.

The precision of the polygonal approximation is controlled by the parameter *error*. The greater is the error value the coarser is the approximation and the less there are polygons.

The output image *im_out* contains a set of 2D lines, each of them use a different color.

Warning: if the contour is not 1 pixel thickness, the operator may have unpredictable behavior. It might be necessary to use the operator ppostthinning to guaranty 1 pixel thickness.

Parameters

- *error* specifies the maximum distance allowed to the original curve. It controls the precision of the polygonalization. The greater is the error value the coarser is the approximation.

Inputs

- *im_in*: a 2D grayscale image (Img2duc) which contains the contours=.

Outputs

- *im_out*: a 2D grayscale image (Img2duc).

Result

Returns SUCCESS or FAILURE.

Examples

Performs a polygonal approximation of the boundaries of tangram pieces:

```
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
```

See also

Contour

C++ prototype

```
Errc PPolygonalApproximation( const Img2duc &im_in, Img2duc &im_out,
int error );
```

Version française

Approximation polygonale des contours d’une image.

Author: Serge Coudé

ppolynomialfitting

Approximates image to flat background using polynomial fitting.

Synopsis

ppolynomialfitting *xorder yorder xyorder* [*im_in*|-] [*im_mk*|-] [*im_out*|-]

Description

ppolynomialfitting converts an image content into a flat background image, using a polynomial fitting.

The image *im_mk* is used as a mask, and defines the list of pixels that can be used to compute the polynomial approximation.

The order of the polynomial can be selected separately for x, y, and mixed terms. For example, with orders 2, 3, and 2 for x, y, and xy respectively, the polynomial will be:

```
a+b*x+c*x^2+d*y+e*y^2+f*y^3+g*xy
```

Parameters

- *xorder* is the x order [0..10].
- *yorder* is the y order [0..10].
- *xyorder* is the xy order [0..10].

Inputs

- *im_in*: a 2D float image.
- *im_mk*: a 2D gray image used a mask image.

Outputs

- *im_out*: a 2D image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

- Correction the illumination of the tangram image using background subtraction. The background is approximated by a polynomial:

```
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
```

More examples

See also

Surface Fitting

C++ Prototype

```
Errc PPolynomialFitting( const Imx2d &im_in, const Img2d &im_mk,
Imx2d &im_out, int xOrder, int yOrder, int xyOrder );
```

Version française

Calcul de l’approximation du fond d’une image en utilisant une approximation polynomiale.

Author: Régis Clouard

ppostthinning

Performs contour postthinning to guaranty 8-connexity (or 26-connexity).

Synopsis

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

Description

ppostthinning consists in deleting all pixels that do not guaranty the 8-connexity in 2D (or the 26-connexity in 3D) for contour chains.
A contour chain is a sequence of connected non null pixels with 1 pixel thickness. There are connected with the 8-connexity in 2D (or the 26-connexity in 3D).

A pixel "x" is deleted from the chain if it does not destroy the 8-connexity (or 26-connexity). For example, the center is deleted in case of the following configuration (and any other symmetrical configurations):

```
  |x|       | |0
x|x|   or x|x|
  | |0      |x|
```

Warning: **ppostthinning** needs contour with 1 pixel thickness. It may be necessary to use before a skeletonization operator, for example: pskeletonization.

The output image *im_out* is of the same type as the input image *im_in*.

Inputs

- *im_in*: a 2D grayscale image (Img2duc) which contains the contours.

Outputs

- *im_out*: a 2D grayscale image (Img2duc).

Result

Returns SUCCESS or FAILURE.

Examples

Extracts contours from the tangram.pan image:

```
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
pgeodesicdilatation 1 1 -1 i6.pan i7.pan out.pan
```

See also

Contour

C++ prototype

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

Version française

Suppression des points de contours superflus.

Author: Régis Clouard

ppow

Computes nth power of an image or a graph.

Synopsis

ppow *n* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

ppow computes the nth power of the input *im_in*.

If *im_in* is an image then the new image *im_out* is built with the power of each pixel:

```
pixel(im_out)=pow(pixel(im_in),n)
```

The output image *im_out* is always a Float image.

For color or multispectral image, the power is computed separately on each band.

If *im_in* is a graph then the new graph *im_out* is built with the nth power of each node value.

Parameters

- *n* is a real and represents the power.

Inputs

- *im_in*: an image or a graph.

Outputs

- *im_out*: a Float image or a graph.

Result

Returns SUCCESS or FAILURE.

Examples

Computes the power 2 of the tangram.pan image.

```
ppow 2 tangram.pan a.pan
```

See also

Arithmetic

C++ prototype

```
Errc PPow( const Img2duc &im_in, Img2duc &im_out, double n );
```

Version française

Puissance nième d’une image ou d’un graphe.

Author: Régis Clouard

ppowerlawtransform

Performs power-law transform of the gray-levels.

Synopsis

ppowerlawtransform *gamma min max* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

ppowerlawtransform expands or compresses gray-levels of the input image using a power law transform of the gray-levels according to the value of the parameter *gamma*. Such transform often refers to the gamma correction.

The effect of the power-law transform is to map a narrow range of low gray-level values in the input image into a wider range of output levels when gamma < 1, and the opposite with gamma >1.

The power-law transform of pixel 'p' has the form:

```
im_out[p]=(c * (im_in[p]-smin)^gamma) + min;
c=(max-min) / (smax-smin)
```

where smin and smax are the minimum and the maximum values of the input image, and c is a normalization factor for stretching output values between *min* and *max*.

For color and multispectral images, the transform uses the vectorial approach: the min and max values are calculed from all the bands, and then each band is stretched with the same transform.

Parameters

- *gamma* is a positive real value. It specifies the degree of transformation. Values < 0 compress low gray-levels while expand high gray-levels. Values >0 expand low gray-levels and compress high gray-levels. A value of gamma=1 leads to a linear transform. Typical values are 0.04, ... , 0.4, 1, 1.5, ... 25.0.

- *min* and *max* specify the bounds of the output pixel value. They are related to the type of the input image.
  Note: if *min* < *max* then min and max are set with the minimum and maximum values of the input image type: for example, 0 and 255 for Uchar images.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image with the same properties as im_in.

Result

Returns SUCCESS or FAILURE in case of invalid parameter values.

Examples

Applies a positive transform followed by a negative transform to create image b.pan. Because negative is the inverse transform of positive transform, b.pan is (almost) equal to tangram.pan (due to rounding error):

```
ppowerlawtransform 2 0 255 tangram.pan a.pan
ppowerlawtransform 0.5 28 165 a.pan b.pan
```

Applies a logarithmic transform to create a.pan and uses the min and max values of the image type as new bounds:

```
ppowerlawtransform 0.4 1 -1 tangram.pan a.pan
```

See also

Lut transform, plineartransform, plogtransform

C++ prototype

```
Errc PPowerLawTransform( const Img2duc &im_in, const Img2duc
&im_out, float gamma, float min, float max );
```

Version française

Transformation des niveaux de gris par une loi de puissance.

Author: Régis Clouard

pppm2pan

Converts PPM (Portable PixMap), PGM (Portable GrayMap) ou PBM (Portable BitMap) image file to Pandore file.

Synopsis

pppm2pan *im_in* [*im_out*|-]

Description

pppm2pan converts a PPM (Portable PixMap image), PGM (Portable GrayMap image), or PBM (Portable BitMap image) to Pandore image file.

A PPM, PGM, PBM consists of the followings:

- The first line contains the "magic number" among:
  - P2: for binary image with ASCII data.
  - P2: for gray scale image with ASCII data.
  - P4: for binary image with pure binary data.
  - P5: for gray scale image with pure binary data.
  - P3: for color image with ASCII data.
  - P6: for color image with pure binary data.
- The second line contains the number of columns and rows in ASCII.
- The third line contains the number of colors.

  Each line can include comments introduced by #.

- The data format depends on the magic number: (Caution: No line should be longer than 70 characters.)
  - P1: pixel is represented as an ASCII decimal number (0: while, 1:black).
  - P2: pixel is represented as an ASCII decimal number.
  - P5: pixel is represented as 1 bit (0: while, 1: black).
  - P5: pixel is represented as 1 byte.
  - P3: pixel is represented as a triplet of ASCII decimal number (Red, Green Blue).
  - P6: pixel is represented as a triplet of bytes (Red, Green Blue).

Examples:

```
P2
#Gray scale image
#with 5 columns and 5 rows.
5 4
255
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

Tip: PPM image can be easily built by hand. This is a convenient way to build image from scratch.

Inputs

- *im_in*: a PPM or PGM image file.

Outputs

- *im_out*: a 2D Pandore image.

Result

Returns SUCCESS or FAILURE.

Examples

```
pppm2pan image.ppm image.pan
```

See also

Conversion, ppan2ppm

C++ prototype

```
Errc PPPM2Pan( const char* filename, Pobject **obj_out );
```

Version française

Conversion d’un fichier de format PPM (Portable PixMap), PGM (Portable GrayMap) ou PBM (Portable BitMap) en un fichier Pandore.

Author: Régis Clouard

pprewitt

Computes the Prewitt gradient magnitude.

Synopsis

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

Description

pprewitt computes an approximation of the gradient magnitude of the input image *im_in*.

The algorithm uses the convolution with the following kernel:

```
|+1 +1 +1|
|+0 +0 +0|
|-1 -1 -1|
```

The kernel is oriented in two directions: 0 and 90 degrees and the magnitude value is set to the maximum value between these two values.

The output image *im_out* is of the same type as the input image *im_in*.

Inputs

- *im_in*: a grayscale image.

Outputs

- *im_out*: an image of the same type as im_in.

Result

Returns SUCCESS or FAILURE.

Examples

Performs an edge detection for the tangram.pan image:

```
pprewitt tangram.pan b.pan
pbinarization 40 1e30 b.pan out.pan
```

See also

Edge detection

C++ prototype

```
Errc PPrewitt( const Img2duc &im_in, Img2duc &ima );
```

Version française

Module du gradient de Prewitt.

Author: Régis Clouard

pproperty

Gets Pandore object attribute value.

Synopsis

pproperty *attribute_number* [*im_in*|-]

Description

pproperty returns the attribute value of the Pandore object. The attribute is specified by a number based on a pure convention (see parameters).

The value can than be get using the operator **pstatus**.

Parameters

- *attribute_number* is based on the following convention:
  - 0: the number of columns (for image, region map and graph);
  - 1: the number of rows (for image, region map and graph);
  - 2: the number of planes (for image, region map and graph);
  - 3: the number of bands for image, or
    the number of labels for region map, or
    the number of nodes for graph (size).
  - 4 : the color space number:
    [-1: no color space (or gray scale), 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].

Inputs

- *im_in*: a Pandore file.

Result

Returns the value of the selected property.

Examples

Unix/Linux/MACOS: Builds a new image with the same dimension than tangram.pan which contains a white disc of radius 50:

```
pproperty 0 tangram.pan; w=‘pstatus‘
pproperty 1 tangram.pan; h=‘pstatus‘
pshapedesign $w $h 0 1 50 50 a.pan
```

MsDos: Builds a new image with the same dimension than tangram.pan which contains a white disc of radius 50:

```
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
```

See also

Information

C++ prototype

```
Errc PProperty( const Img2duc &img, int attribute_number );
```

Version française

Retourne la valeur d’un attribut d’un objet Pandore.

Author: Régis Clouard

ppsnr

Computes the Peak Signal to Noise Ratio.

Synopsis

ppsnr *max* [*im1_in*|-] [*im2_in*|-]

Description

ppsnr measures the Peak Signal to Noise ration between the initial image *im_in1* and the restored or enhanced image *im_in2*.
The Peak Signal to Noise Ratio (PSNR) is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. It is defined via the mean squared error (MSE) between the two input images where *im1_in* is the input image and *im2_in* is the restored or enhanced version of *im1_in*.
Consequently, the higher is the PSNR, the better is the signal and consequently the related image processing (restoration or enhancement).

Because many signals have a very wide dynamic range, PSNR is expressed in terms of decibel (dB). Typical values for the PSNR in image compression are between 30 and 40 dB.

PSNR is defined as follows:

$$PSNR = 10 * \log_{10}((max*max)/MSE);$$
$$\text{with } MSE=sum((im\_in1-im\_in2)^2)/N$$

where *max* is the maximum pixel value of the input images (not greater than the possible maximum value of the input image) and N is the total number of pixels of the input image. If *max*=-1 then max=Max(ims_in1)-Min(im_in1)).

Input images *im_in1* and *im_in2* must have the same dimensions and the same type.

For color images or multispectral images, the definition of PSNR is the same except that the MSE is the sum over all squared value differences divided by image size and by the number of bands (ie. 3 for the color images).

Parameter

- *max* is the maximal pixel value. Typically, max=255 for Uchar images.
  If *max*=-1 then max=Max(ims_in1)-Min(im_in1)).

Inputs

- *im_in1*: an image.
- *im_in2*: an image (restored or enhanced version of im_in1).

Result

Returns the ratio value as a positive real value expressed in dB.
(Use `pstatus` to get this value).

Examples

Adds gaussian noise with mean 0 and standard deviation 1.5 to tangram.pan image and then computes the PSNR for the meanfilter smoothing operator:

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

See also

Evaluation, pmse, psnr

C++ prototype

```
Errc PPSNR( const Img2duc &im_in1, const Img2duc &im_in2, Float max
);
```

Version française

Calcul du rapport signal sur bruit en crête.

Author: Régis Clouard

pqmf

Designs Quadratic Mirror Filter for wavelet transform.

Synopsis

pqmf *name order* [*col_out*|-]

Description

pqmf designs a Quadratic Mirror filter for wavelet transform. This filter can then be used with operator pdwt.

Information stored in the filter is the name of the filter, and the set of lowpass filter coefficients to allows the calculus of the dyadic wavelet decomposition.

Parameters

- *name* specifies the name of the filter among 7 available filters.
- *order* then specifies the order of the filter.
  - 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

Output

- *col_out*: a collection that contains the filter coefficients.

Result

Returns SUCCESS or FAILURE.

Examples

Builds a synthetic image (a square) to illustrate the Gibbs phenomenon in wavelets analysis.

```
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
```

See also

Frequency, dwt, idwt

C++ prototype

```
Errc PQmf( const char *name, const char *order, Collection& col_out
);
```

Version française

Génération d’un filtre QMF pour la transformée en ondelette.

Author: Ludovic Soltys

pquadraticbsplinerescale

Performs a rescaling of image using the Quadratic Bezier Spline interpolation.

Synopsis

pquadraticbsplinerescale *zoomx zoomy zoomyz* [*im_in*|-] [*im_out*|-]

Description

pquadraticbsplinerescale uses a convolution kernel to interpolate the pixel of the input image *im_in* in order to calculate the pixel value of the output image *im_out*. The interpolation consists in weigthing the input pixels influence on the output pixels. The weights are relative to the position of the output pixels and are given by the quadratic Bezier spline function:

```
        | - x*x + 0.75 if -0.5 <= x <= 0.5
Q(x) = | (0.5*x*x - 1.5*x + 1.125) 1.5 <= x <= 1.5
        | 0 otherwise
```

For example, if the image is scaled by 3, then each output pixel is:

```
for i in [-2, 2]
  for j in [-2, 2]
      im_out[p] += Q(i)*Q(j)*im_in[p]
```

To rescale region map or graph, use the operator prescale.

Parameters

- *zoomx*, *zoomy*, *zoomz* are positive real values.
  - if a zoom factor is > 1 then the image is enlarged along the related axis.
  - if a zoom factor is < 1 then the image is shrunk along the related axis.

  (*zoomz* is ignored for 2D images but must be given).

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

- Enlarges the tangram.pan 2D image by a factor 2:

```
pquadraticbsplinerescale 2 2 0 tangram.pan a.pan
```

- Shrinks the tangram.pan 2D image by a factor 2:

```
pquadraticbsplinerescale 0.5 0.5 0 tangram.pan a.pan
```

See also

Transformation, plinearrescale, pbicubicrescale, planczosrescale, pmitchellrescale, prescale

C++ prototype

```
Errc PQuadraticBSplineRescale( const Img2duc &im_in, Img2duc
&im_out, const float zoomy, const float zoomx );
```

Version française

Retaille d’une image par fonction spline quadratique de Bezier.

Author: Régis Clouard

pras2pan

Converts Sun raster image file to Pandore image file.

Synopsis

pras2pan *im_in* [*im_out*|-]

Description

pras2pan converts Sun Raster file to Pandore image file. The result Pandore image type *im_out* depends on the Raster file type *im_in*.

Sun Raster File is always a 2D image.

Inputs

- *im_in*: a Sun Rasterfile image file.

Outputs

- *im_out*: a 2D Pandore image file.

Result

Returns SUCCESS or FAILURE.

Examples

Converts he SunRaster File to Pandore image:

```
pras2pan image.ras image.pan
```

See also

Conversion

C++ prototype

```
Errc PRas2Pan( const Char* filename, Pobject **objout );
```

Version française

Conversion d’une image Rasterfile en une image Pandore.

Author: Régis Clouard

praw2pan

Converts raw image file to Pandore image file.

Synopsis

praw2pan *bytes ncol nrow ndep im_in* [*im_out*|-]

Description

praw2pan converts a binary image to a Pandore image.

Raw images is supposed to be encoded by bands, then by depth then by line and finally by column.

Caution: The raw image is endian system dependent. Use the parameter *bytes* to specify if the endian conversion must be done during the file conversion.

Tip: Because, the data are read from the end of the file, this operator can be used to convert any image type that encodes the image data in binary code. Therefore, any header is skip.

Parameters

- *bytes* specifies the number of bytes for encoding a pixel. By pure convention, a negative value indicates to exchange endian encoding (LSB <->MSN). This parameter also determines the output file format:
  - bytes=1 (o -1) creates a Uchar image.
  - bytes=2 (or -2) creates a Long image.
  - bytes=3 (or -3) creates a Long image.
  - bytes=4 (or -4) creates a Long image.
  - bytes=6 (or -6) creates a Float image (pure convention).
- *ncol*, *nrow* nd *ndep* specify respectively the number of columns, rows and plane of the image. If *ndep*=0 then the image is a 2D image, and if *nrow*=0 then the image is a 1D image.

Inputs

- *im_in*: a regular binary image file.

Outputs

- *im_out*: a Pandore image file.

Result

Returns SUCCESS or FAILURE.

Examples

```
praw2pan image.raw image.pan
```

See also

Conversion, ppan2raw

C++ prototype

```
Errc PRaw2Pan( const char* filename, pobject** obj_out, Long npix,
Long nlig, Long ncol );
```

Version française

Conversion d’un fichier image sans entête en un fichier Pandore.

Author: Régis Clouard

prds

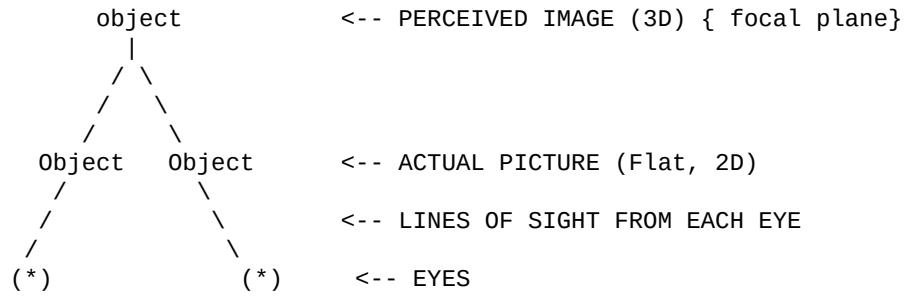
Builds Random Dot Stereogram image.

Synopsis

prds [*im_in*|-] [*im_out*|-]

Description

prds builds a Random Dot Stereogram image from the depth image *im_in*. A stereogram is an image in which stereoscopic information are encoded. To look at a stereogram, one has to consider that the focal plane is behind the image.



The depth image is a gray level image where each pixel value is set with the depth of the related point in the scene. Values belongs to [0..255] where 0 is the minimum distance and 255 the maximum distance. The depth image can be built from the projection of a 3D image to 2D image.

The result image is apparently a noise image but the 3D scene is hidden inside the image.

Inputs

- *im_in*: a 2D image of bytes (Img2duc). Each pixel encodes the distance from the observer of the related point in the scene.

Outputs

- *im_out*: a 2D image of bytes (Img2duc).

Result

Returns SUCCESS or FAILURE in case of bad input.

Result

Builds a Random Dot Stereogram from the 3D image cyto3d.pan:

```
pdepth2graylevel 50 cyto3d.pan i0.pan
pmultcst 10 i0.pan i1.pan
prds i1.pan rds_out.pan
```

See also

Miscellaneous, pstereogram

C++ prototype

```
Errc PRds( const Img2duc &imp, Img2duc &im_out );
```

Version française

Construction d’une image stéréogramme type Random Dot Stereogram.

Author: Régis Clouard

prectangularityselection

Selects regions from rectangularity degree.

Synopsis

prectangularityselection *relation threshold* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

prectangularityselection selects regions from their rectangularity degree. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

Rectangularity is the ratio of a region area to the area of the minimum bounding rectangle. The maximum value is 1.0 for a circle.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is a real value from [0..1] which is the rectangularity degree.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *rg_out*: a 2D region map.

Result

Returns the number of selected regions.

Examples

Selects regions that have a rectangularity degree > 10:

```
prectangularityselection 1 10 in.pn rout.pan
```

See also

Region

C++ prototype

```
Errc PRectangularitySelection( const Reg2d &rg_in, Reg2d &rg_out,
int relation, float threshold );
```

Version française

Sélection de régions sur leur valeur de rectangularité.

Author: Régis Clouard

pregionarea

Measures region area size.

Synopsis

pregionarea *name* [*-m mask*] [*rg_in*|-] [*col_out*|-]

Description

pregionarea measures the area size of the regions inside the region map *rg_in*. Each region area size is stored as a unsigned long value in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

The area size is calculated from the number of pixels included in the region and on the boundary. The algorithm uses one half pixel for concavity. For example, the area is 10 pixels for the following region (`8 + 4*0.5`):

```
 xx
xxxx
 xx
```

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays area size of the regions yielded by a simple binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionarea area b.pan c.pan
pcol2txt c.pan
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionSurface( const Reg2d &rg_in, Collection &cold, const
std::string &name );
```

Version française

Calcul de la surface des régions.

Author: Alexandre Duret-Lutz

pregioncompactness

Measures region compactness factor.

Synopsis

pregioncompactness *name* [*rg_in*|-] [*col_out*|-]

Description

pregioncompactness measures compactness of the regions inside the region map *rg_in*. Each region compactness is stored as a float in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

Compactness represents the degree to which the shape region is compact. It is defined as the ratio of the area of a region to the area of a circle with the same perimeter. It is calculated as follows:

```
compactness = (4*PI*area) / (perimeter*perimeter)
```

For a circle, the compactness is 1.0, for a square, it is PI/4 and for an infinitely long and narrow shape, it is zero.

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays compactness of the regions yielded by a simple binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregioncompactness compactness b.pan c.pan
pcol2txt c.pan
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionCompactness( const Reg2d &rg_in, Collection &cold, const
std::string &name );
```

Version française

Calcul de la compacité des régions.

Author: Alexandre Duret-Lutz

pregionconvexity

Measures region convexity degree.

Synopsis

pregionconvexity *name* [*rg_in*|-] [*col_out*|-]

Description

pregionconvexity measures the convexity degree of the regions inside the region map *rg_in*. Each region convexity degree is stored as a float in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

Convexity is the relative amount that a region differs from a convex region. It is calculated as follows:

```
convexity = region area / convex hull area.
```

The maximum value is 1.0 for convex region (eg. circle, square).

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays convexity of the regions yielded by a simple binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionconvexity convexity b.pan c.pan
pcol2txt c.pan
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionConvexity( const Reg2d &rg_in, Collection &cold, const
std::string &name );
```

Version française

Calcul de la convexité des régions.

Author: Alexandre Duret-Lutz

pregioneccentricity

Measures region eccentricity degree.

Synopsis

pregioneccentricity *name* [*rg_in*|-] [*col_out*|-]

Description

pregioneccentricity measures the eccentricity degree of the regions inside the region map *rg_in*. Each region eccentricity degree is stored as a float in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

Eccentricity measures how much the region deviates from being circular. It is defined as the ratio of the length of the short axis to the length of the long axis:

$$\text{eccentricity} = \frac{(Mxx+Myy-\sqrt{(Mxx-Myy)*(Mxx-Myy)+4*Mxy*Mxy})}{(Mxx+Myy+\sqrt{(Mxx-Myy)*(Mxx-Myy)+4*Mxy*Mxy})}$$

The maximum value is 1.0 for a square or a circle.

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays eccentricity of the regions yielded by a simple binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregioneccentricity eccentricity b.pan c.pan
pcol2txt c.pan
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionEccentricity( const Reg2d &rg_in, Collection &cold,
const std::string &name );
```

Version française

Calcul de l’excentricité des régions.

Author: Alexandre Duret-Lutz

pregionelongation

Measure region elongation factor.

Synopsis

pregionelongation *name* [*rg_in*|-] [*col_out*|-]

Description

pregionelongation measures the elongation factor of the regions inside the region map *rg_in*. Each region elongation factor is stored as a float in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

Elongation is the ratio between the length and the width of the bounding box:

`elongation = width(bounding box)/width(bounding box).`

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays elongation of the regions yielded by a simple binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionelongation elongation b.pan c.pan
pcol2txt c.pan
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionElongation( const Reg2d &rg_in, Collection &cold, const
std::string &name );
```

Version française

Calcul de l’élongation des régions.

Author: Alexandre Duret-Lutz

pregionenergy

Measures region energy value.

Synopsis

pregionenergy *name* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*col_out*|-]

Description

pregionenergy measures the energy value of the regions inside the region map *rg_in*. Each region energy value is stored as a float in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

Energy is measure as follows:

```
energy = SUM { im_in[p] * im_in[p]} / N
```

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays energy of the regions yielded by a simple binarization of 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
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionEnergy( const Reg2d &rg_in, const Img2duc &im_in,
Collection &cold, const std::string &name );
```

Version française

Calcul de l’énergie des régions.

Author: Alexandre Duret-Lutz

pregioneulernumber

Measures region Euler number.

Synopsis

pregioneulernumber *name* [*rg_in*|-] [*col_out*|-]

Description

pregioneulernumber measures the Euler number of the regions inside the region map *rg_in*. Each region Euler number is stored as a long value in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

The Euler number for a region is defined as 1 - the number of holes in the region.

The algorithm used to calculated the Euler number rests on the local operation:
Let X(R) the number of the following 2x2 patterns (r region label for region R, et 0 other labels):

```
0 0
0 r
```

Let V(R) the number of the following 2x2 pattern

```
0 r
r r
```

then Euler(R) = X(R) - V(R)

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays euleur number of the regions yielded by a simple binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregioneulernumber eulernumber b.pan c.pan
pcol2txt c.pan
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionEulerNumber( const Reg2d &rg_in, Collection &cold, const
std::string &name );
```

Version française

Calcul du nombre d’Euler des régions.

Author: Régis Clouard

pregionmaximum

Measures region maximum grayscale value.

Synopsis

pregionmaximum *name* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*col_out*|-]

Description

pregionmaximum measures the maximum grayscale value of the regions inside the region map *rg_in*. Each region maximum grayscale value is stored as a float in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays mean of the regions yielded by a simple binarization of 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
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionMaximum( const Reg2d &rg_in, const Img2duc &im_in,
Collection &cold, const std::string &name );
```

Version française

Calcul de la valeur maximale des régions.

Author: Régis Clouard

pregionmean

Measures region mean grayscale value.

Synopsis

pregionmean *name* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*col_out*|-]

Description

pregionmean measures the mean grayscale value of the regions inside the region map *rg_in*. Each region mean grayscale value is stored as a float in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

The mean value is calculated as follows:

```
mi= SUM(im_in[p] / p in Ri) / N
```

where N is the number of pixels in the region.

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a region map.
- *im_in*: an image.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays mean of the regions yielded by a simple binarization of 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
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionMean( const Reg2d &rg_in, const Img2duc &im_in,
Collection &cold, const std::string &name );
```

Version française

Calcul de la moyenne des régions.

Author: Alexandre Duret-Lutz

pregionminimum

Measures region minimum grayscale value.

Synopsis

pregionminimum *name* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*col_out*|-]

Description

pregionminimum measures the minimum grayscale value of the regions inside the region map *rg_in*. Each region minimum grayscale value is stored as a float in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays minimum of the regions yielded by a simple binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionminimum minimum b.pan tangram.pan c.pan
pcol2txt c.pan
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionMinimum( const Reg2d &rg_in, const Img2duc &im_in,
Collection &cold, const std::string &name );
```

Version française

Calcul de la valeur minimale des régions.

Author: Régis Clouard

pregionorientation

Measures region orientation degree.

Synopsis

pregionorientation *name* [*rg_in*|-] [*col_out*|-]

Description

pregionorientation measures the orientation of the regions inside the region map *rg_in*. Each region orientation is stored as a unsigned long value in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

The orientation value is specified in degree. Is is computed from the moments:

`orientation = 0.5*arctan(2*M11/(M20-M02)).`

If `M20=M02` the region is symmetrical and orientation is set to 360.

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays orientation of the regions yielded by a simple binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionorientation orientation b.pan c.pan
pcol2txt c.pan
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionOrientation( const Reg2d &rg_in, Collection &cold, const
std::string &name );
```

Version française

Calcul de l’orientation des régions.

Author: Alexandre Duret-Lutz

pregionperimeter

Measures region perimeter length.

Synopsis

pregionperimeter *name* [*rg_in*|-] [*col_out*|-]

Description

pregionperimeter measures the perimeter length of the regions inside the region map *rg_in*. Each region perimeter length is stored as a unsigned long value in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

The region perimeter is the number of pixels on the region boundary. The algorithm uses one quarter pixel for concavity. For example, the perimeter is 7 pixels for the following region (`7=6+4*0.25`):

```
 xx
xxxx
 xx
```

For a region with 1 pixel, the perimeter is 1.

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays perimeter length of the regions yielded by a simple binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionperimeter perimeter b.pan c.pan
pcol2txt c.pan
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionPerimeter( const Reg2d &rg_in, Collection &cold, const
std::string &name );
```

Version française

Calcul du périmètre des régions.

Author: Alexandre Duret-Lutz

pregionrectangularity

Measures region rectangularity degree.

Synopsis

pregionrectangularity *name* [*-m mask*] [*rg_in*|-] [*col_out*|-]

Description

pregionrectangularity measures the rectangularity of the regions inside the region map *rg_in*. Each region rectangularity is stored as a float in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

Rectangularity is the ratio of a region area to the area of the minimum bounding rectangle. The maximum value is 1.0 for a square.

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays rectangularity of the regions yielded by a simple binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionrectangularity rectangularity b.pan c.pan
pcol2txt c.pan
```

See also

Region Features Extraction

C++ prototype

```
PRegioinRectangularity( const Reg2d &rg_in, Collection &cold, const
std::string &name );
```

Version française

Calcul de la rectangularité des régions.

Author: Alexandre Duret-Lutz

pregionsphericity

Measures region sphericity degree.

Synopsis

pregionsphericity *name* [*rg_in*|-] [*col_out*|-]

Description

pregionsphericity measures the sphericity of the regions inside the region map *rg_in*. Each region sphericity is stored as a float in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

Sphericity measures the degree to which an object approaches the shape of a sphere. It is calculated using:

```
sphericity = rayon_inscribing / rayon_circumscribing.
```

The maximum value is 1.0 for circle.

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays sphericity of the regions yielded by a simple binarization of tangram.pan:

```
pbinarization 100 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pregionsphericity sphericity b.pan c.pan
pcol2txt c.pan
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionSphericity( const Reg2d &rg_in, Collection &cold, const
std::string &name );
```

Version française

Calcul de la sphéricité des régions.

Author: Régis Clouard

pregionvariance

Measures region variance.

Synopsis

pregionvariance *name* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*col_out*|-]

Description

pregionvariance measures the internal variance of the regions inside the region map *rg_in*. Each region variance is stored as a float in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

The variance for the region i is calculated as follows:

```
var = ((n*sigma2)-(sigma*sigma)) / (N*N)
```

where *sigma* is the sum of the gray levels inside the region i.
where *sigma2* is the sum of the square of the gray levels inside the region i;
where *N* is the number of pixels inside the region i.

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays variance of the regions yielded by a simple binarization of 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
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionVariance( const Reg2d &rg_in, const Img2duc &im_in,
Collection &cold, const std::string &name );
```

Version française

Calcul de la variance des régions.

Author: Alexandre Duret-Lutz

pregionvolume

Measures region volume.

Synopsis

pregionvolume *name* [*-m mask*] [*rg_in*|-] [*col_out*|-]

Description

pregionvolume measures the volume of the regions inside the region map *rg_in*. Each region volume is stored as a unsigned long value in the array named *name* inside the collection *col_out*. The ith item in the array corresponds to the i+1th region because the region 0 is not considered.

The volume is calculated in pixel unit.

Parameters

- *name* is a string that contains the name of the array inside the collection.

Inputs

- *rg_in*: a 3D region map.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Displays volume of the regions yielded by a simple binarization of 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
```

See also

Region Features Extraction

C++ prototype

```
Errc PRegionVolume( const Reg3d &rg_in, Collection &cold, const
std::string &name );
```

Version française

Calcul de volume des régions.

Author: Alexandre Duret-Lutz

pregistrationPDE

Computes the displacement field between two input images.

Synopsis

pregistrationPDE *smoothness precision* [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

pregistrationPDE estimates the 2D displacement field *im_out* between the two images *im_in1* and *im_in2*. The field is calculated from *im_in1* to *im_in2*.

The displacement field is based on the minimization of the following criteria:

```
E(U) = integral(|I1(p)-I2(p+U)| + smoothness*Laplac(U)).
```

Such minimization is done by PDE descent with different image scales.

The result is a multispectral image with two bands. The first band corresponds to x-component of the displacement vector and the second band to the y-component. This vector field can be drawn with the operator pplotquiver.

Parameters

- *smoothness* is a real value between [0..1] which controls the regularity of the field. The value 0 corresponds to no regularity, 0.1 to an average regularity and 0.9, to a strong regularity (quasi-constant field). If the displacement is known as rigid (eg. translation) a high smoothness value is recommended. When the displacement is known as non rigid a low smoothness value is recommended.
- *precision* is a real value which defines the precision factor. The higher is the precision, the longer is the computing time. A typical value is 0.9.

Inputs

- *im_in1* a 2D image.
- *im_in1* a 2D image (same type and same size as im_in1).

Outputs

- *im_out* a 2D multispectral image.

Result

Returns SUCCESS or FAILURE.

Examples

Displays the vector field of a translation of the tangram.pan image:

```
ptranslation 0 10 tangram.pan tangram1.pan
pregistrationPDE 0.1 0.9 tangram.pan tangram1.pan a.pan
pplotquiver 800 800 10 -20 a.pan out.pan
```

See also

Motion

C++ prototype

```
Errc PRegistrationPDE( const Imx2d &ims1, const Imx2d &ims2, Img2duc
&imd, Float smoothness, Float precision );
```

Version française

Estimation du champ de déplacement entre deux images.

Important notice

Author: D. Tschumperlé

prelabelingfromarray

Relabels regions from label array.

Synopsis

prelabelingfromarray *name* [*col_in*|-] [*rg_in*|-] [*rg_out*|-]

Description

prelabelingfromarray relabels region of the input region map *rg_in* from the values of the array named *name* given in the input collection *col_in*.

The label of the input region *x > 0* is set with the new label *name[x-1]*. If the array *name* has less value than the label values then the remainder regions are discarded from the result region map. The region 0 is not relabeled.

Inputs

- *col_in*: a collection.
- *rg_in*: a region map.

Outputs

- *rg_out*: a region map.

Parameters

- *name* specifies the name of the array in the collection.

Result

Returns the maximum label value of the output region map.

Examples

Relabels regions from the vector foo in the collection col.pan:

```
prelabelingfromarray foo col.pan rin.pn rout.pan
```

See also

Region

C++ prototype

```
Errc PRelabelingFromArray( const std::string &name, const Collection
&in, const Reg2d &reg_in, Reg2d &reg_out );
```

Version française

Relabelisation d’une carte de régions à partir des valeurs d’un vecteur d’etiquettes.

Author: Alexandre Duret-Lutz

prelabelingwithgraph

Relabels region and related graph node.

Synopsis

prelabelingwithgraph [*rg_in*|-] [*gr_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

prelabelingwithgraph relabels regions of the input region map *rg_in* and the related nodes in the input graph *gr_in*. Each region of the output region map is set with a new label so as to use all labels between 0 and the number of regions. The related graph nodes of the output graph *graph_out* are set with the same label value.

Remarque: To relabel a region map without a graph use operator plabeling.

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.

Outputs

- *rg_in*: a region map.
- *gr_out*: a graph.

Result

Returns the maximum label value in the region map.

Examples

Relabels regions of the rin.pan region map and updates the related graph g.pan:

```
relabelingwithgraph rin.pan g.pan rut.pn
```

See also

Region, plabeling

C++ prototype

```
Errc PRelabelingWithGraph( const Reg2d &rg_in, Graph2d &gr_in, Reg2d
&rg_out, Graph2d &gr_out );
```

Version française

Renumérotation des régions d’une carte et des sommets du graphe associé.

Author: Régis Clouard

premoveslice

Removes slice to 3D image.

Synopsis

premoveslice *direction* [*-m mask*] [*im_in1*|-] [*im_out1*|-] [*im_out2*|-]

Description

premoveslice removes a 2D image from the end or the beginning of a 3D image. The 2D image *im_out2* is extracted from the beginning if the parameter *direction* is negative or from the end if it is positive. The new image *im_out1* has one slice less than the input image *im_in1*.

The last 3D image can be casted to 2D image with operator pim3d22d.

The result image *im_out* is of the same type as the two input images.

For region map, it may be judicious to relabel the regions (see plabeling).

Parameters

- *direction* specifies whether the 2D image is removed from the beginning (*direction* < 0) or from the end (*direction* > 0) of the 3D image.

Inputs

- *im_in*: a 3D image or a 3D region map.

Outputs

- *im_out1*: a 3D image or a 3D region map of the same type as the input image.
- *im_out2*: a 2D image or a 2D region map.

Result

Returns SUCCESS or FAILURE in case of bad input or when 3D image is one slice depth.

Examples

Removes a2d.pan to the end of the a3d.pan:

```
premoveslice 1 a3d.pan b3d.pan a2d.pan
```

See also

Utility, pgetslice, paddslice, pim3d22d

C++ prototype

```
Errc PRemoveSlice( const Imx3d &im_in1, const Imx2d &im_out1, Imx3d
&im_out2, int direction );
```

Version française

Suppression d’un plan 2D dans une image 3D.

Author: Régis Clouard

prescale

Performs an affine rescaling of image, region map or graph using the nearest neighbor interpolation.

Synopsis

prescale *zoomx zoomy zoomyz* [*im_in*|-] [*im_out*|-]

Description

prescale changes magnification of the input image by a factor *zoomx* along the x axis, *zoomy* along the y axis and *zoomz* along the z axis (for 3D images). The image is enlarged along an axis if the zoom factor is > 1 and is shrunk if the zoom factor is >0 and <1.

This version uses the nearest neighbor interpolation. Thus, the image shrinking consists in a subsampling of the pixels and the image enlargement consists in a replication of the pixels (This has the effect of simply making each pixel bigger):

```
im_out[z][y][x]=im_in[z/zoomz][y/zoomy][x/zoomx];
```

The nearest neighbor interpolation is the most basic and requires the least processing time of all the interpolation algorithms. However, this results in blocky artifacts within the image.

Better results can be obtained with other rescaling operators such as plinearrescale or pbicubicrescale.

Parameters

- *zoomx*, *zoomy*, *zoomz* are positive real values.

  if a zoom factor is > 1 then the image is enlarged along the related axis.

  if a zoom factor is < 1 then the image is shrunk along the related axis.
  zoomz is ignored for 2D images but must be given.

Inputs

- *im_in*: an image, a region map, or a graph.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

- Enlarges the tangram.pan 2D image by a factor 2:

```
prescale 2 2 0 tangram.pan a.pan
```

- Shrinks the tangram.pan 2D image by a factor 2:

```
prescale 0.5 0.5 0 tangram.pan a.pan
```

See also

Transformation, plinearrescale, pbicubicrescale

C++ prototype

```
Errc Prescale( const Img2duc &im_in, Img2duc &im_out, const float
zoomy, const float zoomx );
```

Version française

Augmentation ou réduction de la taille d’une image par interpolation selon le plus proche voisin.

Author: Régis Clouard

presizing

Performs an affine resizing of image or region map.

Synopsis

presize *width height depth* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

presize resizes the input image *im_in* so has to have the new specified size *depth* x *height* x *width*. The resizing uses an affine transformation. Thus, it might be necessary to use a smoothing operator either before resizing in case of image shrinking or after resizing in case of image magnifying.

Parameters

- *depth, height, width* are integer values. *Depth* is ignored for 2D image.

Inputs

- *im_in*: an image or a region map.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Resizes tangram.pan so has to be a 512x256 image:

```
pmeanfiltering 1 tangram.pan a.pan
presize 512 256 0 a.pan b.pan
```

See also

Transformation

C++ prototype

```
Errc PResize( const Img2duc &im_in, Img2duc &im_out, int height, int
width );
```

Version française

Ajustement de la taille d’une image en fonction d’une taille souhaitée.

Author: Régis Clouard

prg2gr

Converts region map to graph.

Synopsis

prg2gr [*-m mask*] [*rg_in*|-] [*gr_out*|-]

Description

prg2gr builds a graph from a region map, where two neighbour regions in the region map *reg_in* leads to two linked nodes in the output graph *gr_out*. The node is located at centre of mass of the related region. Therefore, the node is not necessarily located into the region.

Each links is built with the default weight 1.0.

Inputs

- *rg_in*: a region map.

Outputs

- *gr_out*: a graph.

Result

Returns SUCCESSS or FAILURE.

Examples

Builds the graph g.pan with the region map r.pan

```
prg2gr r.pan g.pan
```

See also

Casting

C++ prototype

```
Errc PRg2Gr( const Reg2d &rg_in, Graph &gr_out );
```

Version française

Création d’un graphe d’un voisinage à partir d’une carte de régions.

Author: François Angot

prg2im

Converts region map to signed long image.

Synopsis

prg2im [*-m mask*] [*rg_in*|-] [*im_out*|-]

Description

prg2im creates a label image from a region map. Each label is converted to the related grayscale value; for example the label 10 is converted to the signed long value 10.

The output image *im_out* is a signed long image.

Inputs

- *rg_in*: a region map.

Outputs

- *im_out*: a signed long grayscale image (Img2dsl or Img3dsl).

Result

Returns SUCCESS or FAILURE.

Examples

Builds the label image out.pan from a region map built from thresholding of tangran.pan.

```
pthresholding 100 1e30 tangram.pan a.pan
plabeling 8 .pan b.pan
prg2im b.pan out.pan
```

See also

Casting

C++ prototype

```
Errc PRg2Im( const Reg2 &rg_in, Img2dsl &im_out );
```

Version française

Récupération de l’image d’étiquettes d’une carte de régions.

Author: Régis Clouard

prgb2ast

Converts rgb color image to AST color image.

Synopsis

prgb2ast [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prvb2ast converts color image from the color space RGB (Red, Green, Blue) to the color space AST.

AST is the color space defined by Chassery. It is a simplified model based on the components:

$$A = \frac{\log(R) + \log(V) + \log(B)}{3}$$

$$C_1 = \sqrt{3}/2 * (\log(R) - \log(G))$$

$$C_2 = \log(B) - 1/2*(\log(R)+\log(G))$$

A is an achromatic component and C1 and C2 is two chromatic components. From the model, it is possible to compute the hue h and the saturation s:

$$s = \sqrt{C1^2 + C2^2}$$

$$h = \arccos(C1/s)$$

Inputs

- *im_in*: a RGB color image.

Outputs

- *im_out*: a AST color image.

Result

Returns SUCCESS or FAILURE.

Examples

```
prgb2ast parrot.pan a.pan
```

See also

Color

C++ prototype

```
Errc PRGB2AST( const Imc2duc &im_in, Imc2duc &im_out );
```

Version française

Changement d’espace couleur de RGB vers AST.

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.

Author: Olivier Lezoray

prgb2gray

Converts RGB color image to gray space image.

Synopsis

prgb2gray *r g b* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prgb2gray converts a color image of color space RGB to a gray scale image.

The algorithm is as follows:

```
pixel(im_out) = r*pixel.X(im_in)+g*pixel.Y(im_in)+b*pixel.Z(im_in)/(r+g+b);
```

For example, the NTSC standard method uses the following values for r,g and b:

```
r=0.299; g=0.587; b=0.114;
```

Parameters

- *r* specifies the ratio of the Red component.
- *g* specifies the ratio of the Green component.
- *b* specifies the ratio of the Blue component.

Inputs

- *im_in*: a RGB color image.

Outputs

- *im_out*: a color space image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts parrot.pan to gray level image using the NTSC standard conversion:

```
prgb2gray 0.299 0.587 0.114 parrot.pan a.pan
```

See also

Color

C++ prototype

```
Errc PRGB2Gray( const Imc2duc &im_in, Img2dsf &im_out, float r,
float g, float b );
```

Version française

Changement d’espace couleur de RGB vers niveaux de gris.

Author: Régis Clouard

prgb2hsi

Converts RGB color image to HSI color image.

Synopsis

prgb2hsi [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prgb2hsi converts color image from the color space RGB (Red, Green, Blue) to the color space HSI (Hue, Saturation, Intensity).

A hue refers to the gradation of color within the visible spectrum, or optical spectrum, of light. It is expressed in degree unit [0..360].

Saturation or purity is the intensity of a specific hue: a highly saturated hue has a vivid, intense color, while a less saturated hue appears more muted and gray. With no saturation at all, the hue becomes a shade of gray. It is expressed as percentage [0..100].

Lightness is the amount of light in a color. It is expressed in gray level unit [0..255].

The output image is a float image.

The transformation from RGB to HSI is:

```
          [         ((R-G)+(R-B))           ]
H = acos  [ ------------------------------  ]
          [ 2*sqrt((R-G)*(R-G) + (R-B)(G-B)) ]
S = 1 - 3*MIN(R,G,B)
        ------------
        (R + G + B)
I = (R + G + B)/3
```

Thus, primaries RGB color have the followings <H,S,L> values:

Rouge: < 0 , 1 , 85>
Vert: <120 , 1 , 85>
Bleu: <240, 1 , 85> Noir: <90, 1 , 0>

Inputs

- *im_in*: a RGB color image.

Outputs

- *im_out*: a HSI color image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts parrot.pan from rgb to hsi and conversely.

```
prgb2hsi parrot.pan a.pan
phsitogb a.pan b.pan
```

See also

Color, phsi2rgb

C++ prototype

```
Errc PRGB2HSI( const Imc2duc &im_in, Imc2dsf &im_out );
```

Version française

Changement d’espace couleur de RGB vers HSI.

Author: Olivier Lezoray

prgb2hsl

Converts RGB color image to HSL color image.

Synopsis

prgb2hsl [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prgb2hsl converts color image from the color space RGB (Red, Green, Blue) to the color space HSL (Hue, Saturation, Luminosity).

A hue refers to the gradation of color within the visible spectrum, or optical spectrum, of light. It is expressed in degree unit [0..360].

Saturation or purity is the intensity of a specific hue: a highly saturated hue has a vivid, intense color, while a less saturated hue appears more muted and gray. With no saturation at all, the hue becomes a shade of gray. It is expressed as percentage [0..100].

Lightness is the amount of light in a color. It is expressed in gray level unit [0..255].

The output image is a float image.

The transformation from RGB to HSL is:

```
  let max = MAX(R,G,B) and 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)


     (max+min)
L=   --------
         2

    |- 0                 if max = min
    |
    |       max-min
    |- 100 *-------      if l<=1/2
S = |       max+min
```

```
|
|         max-min
|- 100 * ----------- if l>1/2
|        2-(max+min)
```

Thus, primaries RGB color have the followings <H,S,L> values:

Rouge: < 0 , 1 , 85>
Vert: <120 , 1 , 85>
Bleu: <240, 1 , 85> Noir: <90, 1 , 0>

Inputs

- *im_in*: a RGB color image.

Outputs

- *im_out*: a HSL color image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts parrot.pan from rgb to hsl and conversely.

```
prgb2hsl parrot.pan a.pan
phsltogb a.pan b.pan
```

See also

Color, phsl2rgb

C++ prototype

```
Errc PRGB2HSL( const Imc2duc &im_in, Imc2dsf &im_out );
```

Version française

Changement d’espace couleur de RGB vers HSL.

Author: RÃ©gis Clouar

prgb2i1i2i3

Converts RGB color image to (i1,i2,i3) color image.

Synopsis

prgb2i1i2i3 [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prgb2i1i2i3 converts color image from the color space RGB (Red, Green, Blue) to the color space i1,i2,i3.

Inputs

- *im_in*: a color image of format RGB.

Outputs

- *im_out*: a float color image of format i1i2i3.

Result

Returns SUCCESS or FAILURE.

Examples

```
prgb2i1i2i3 parrot.pan a.pan
```

See also

Color

C++ prototype

```
Errc PRGB2I1I2I3( const Imc2duc &Ims, Imc2dsf &Imd );
```

Version française

Changement d’espace couleur RVB vers (i1,i2,i3).

Author: Olivier Lezoray

prgb2pca

Converts rgb color image to principal components.

Synopsis

prgb2pca [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prvb2pca converts color image from the color space RGB (Red, Green, Blue) to principal components by using the Karhunen-Loeve transformation.

Inputs

- *im_in*: a RGB color image.

Outputs

- *im_out*: a PCA color image.

Result

Returns SUCCESS or FAILURE.

Examples

```
prgb2pca parrot.pan a.pan
```

See also

Color

C++ prototype

```
Errc PRGB2PCA( const Imc2duc &im_in, Imc2duc &im_out);
```

Version française

Calcul des composantes principales d’une image couleur.

Author: Olivier Lezoray

prgb2rngnbn

Converts RGB color image to normalized RGB color image.

Synopsis

prgb2rngnbn [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prgbrngnbn converts color image from the color space RGB to the normalized version of RGB.

Each color component is divided by the sum of the three components:

```
red(im_out)=red(im_in)/(red(im_in)+green(im_in)+blue(im_in))
green(im_out)=green(im_in)/(red(im_in)+green(im_in)+blue(im_in))
blue(im_out)=blue(im_in)/(red(im_in)+green(im_in)+blue(im_in))
```

Inputs

- *im_in*: a RGB color image.

Outputs

- *im_out*: a RGB color image.

Result

Returns SUCCESS or FAILURE.

Examples

```
prgb2rngnbn parrot.pan a.pan
```

See also

Color

C++ prototype

```
Errc PRGB2RNGNBN( const Imc2duc &im_in, Imc2dsf &im_out );
```

Version française

Changement d’espace couleur de RGB vers RGB normalisé.

Author: Olivier Lezoray

prgb2wry

Converts RGB color space to (wb,rg,yb) color space.

Synopsis

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

Description

prgb2wry converts color image from the color space RGB (Red, Green, Blue) to the color space (wb,rg,yb) defined par Swain & Ballard.

Inputs

- *im_in*: a color image of RGB format.

Outputs

- *im_out*: a float color image of WRY format.

Result

Returns SUCCESS or FAILURE.

Examples

```
prgb2wry parrot.pan a.pan
```

See also

Color

C++ prototype

```
Errc PRGB2WRY( const Imc2duc &im_in, Imc2duc &im_out );
```

Version française

Changement d'espace couleur de RGB à (wb,rg,yb).

Author: Olivier Lezoray

prgb2xyz

Converts RGB color image to XYZ color image.

Synopsis

prgb2xyz *primaries* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prgb2xyz converts color image from the color space RGB (Red, Green, Blue) to the color space XYZ. Within the XYZ colorspace, each value is represented as a set of positive values from 0..1.

The conversion uses the conversion matrix. For example, for the case *primaries* = 4 (illuminant C primaries NTSC):

```
|  0.607   0.174   0.200 |
|  0.299   0.587   0.114 |
|  0.000   0.066   1.116 | / VALMAX
```

where VALMAX is the maximum value (eg. 255 for bytes images).

Parameters

- *primaries* is an integer from [0..6] which defines the type of 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

Inputs

- *im_in*: a RGB color image.

Outputs

- *im_out*: a XYZ color image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts parrot.pan from rgb to xyz and conversely.

```
prgb2xyz 4 parrot.pan a.pan
pxyz2rgb 4 a.pan b.pan
```

See also

Color

C++ prototype

```
Errc PRGB2XYZ( const Imc2duc &im_in, Imc2dsf &im_out, int primaries
);
```

Version française

Changement d’espace couleur de RGB vers XYZ.

Author: Olivier Lezoray

prgb2ycbcr

Converts RGB color image to YCbCr color image.

Synopsis

prgb2ycbcr [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prgb2ycbcr converts color image from the color space RGB (Red, Green, Blue) the to color space YCbCr.

Inputs

- *im_in*: a RGB color image.

Outputs

- *im_out*: a YCbCr color image.

Result

Returns SUCCESS or FAILURE.

Examples

```
prgb2ycbcr parrot.pan a.pan
```

See also

Color

C++ prototype

```
Errc PRGB2YCBCR( const Imc2duc &im_in, Imc2dsf &im_out );
```

Version française

Changement d’espace couleur de RVB vers YCbCr.

Author: Olivier Lezoray

prgb2ych1ch2

Converts RGB color space to YCh1Ch2 color image.

Synopsis

prgb2ych1ch2 [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prgb2ych1ch2 converts color image from the color space RGB (Red, Green, Blue) the to color space YCh1Ch2 (System of Carron).

Inputs

- *im_in*: a RGB color image.

Outputs

- *im_out*: a YCh1Ch2 color image.

Result

Returns SUCCESS or FAILURE.

Examples

```
prgb2ych1ch2 parrot.pan a.pan
```

See also

Color

C++ prototype

```
Errc PRGB2YCH1CH2( const Imc2duc &im_in, Imc2dsf &im_out );
```

Version française

Changement d’espace couleur de RGB vers YCh1Ch2.

Author: Olivier Lezoray

prgb2yiq

Converts RGB color image to YIQ color image.

Synopsis

prgb2yiq [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prgb2yiq converts color image from the color space RGB (Red, Green, Blue) to the color space YIQ.

YIQ is formerly used in the NTSC television standard. The Y component represents the luma information, and is the only component used by black-and-white television receivers. I and Q represent the chrominance information. In YUV, the U and V components can be thought of as X and Y coordinates within the colorspace.

The conversion uses the linear transformation:

```
Y =     0.299Red    + 0.587Green        + 0.114Blue
I =     0.595716Red - 0.274453Green     - 0.321263Blue
Q =     0.211456Red - 0.522591Green     + 0.311135Blue
```

Inputs

- *im_in*: a RGB color image.

Outputs

- *im_out*: a YIQ color image.

Result

Returns SUCCESS or FAILURE.

Examples

```
prgb2yiq a.pan b.pan
```

See also

Color

C++ prototype

```
Errc PRGB2YIQ( const Imc2duc &im_in, imc2dsf &im_out );
```

Version française

Changement d’espace couleur de RVB vers YIQ.

Author: Olivier Lezoray

prgb2yuv

Converts RGB color image to YUV color image.

Synopsis

prgb2yuv [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

prgb2yuv converts color image from the color space RGB (Red, Gren, Blue) to the color space YUV (television standard).

YUV is adopted by the PAL television standard.

The color conversion is a linear transformation:

```
|  0.299   0.587   0.114 |
| -0.147  -0.289   0.437 |
|  0.615  -0.515  -0.10 |
```

Inputs

- *im_in*: a RGB color image.

Outputs

- *im_out*: a YUV color image.

Result

Returns SUCCESS or FAILURE.

Examples

```
prgb2yuv parrot.pan a.pan
```

See also

pyuv2rgb, Color

C++ prototype

```
Errc PRGB2YUV( const Imc2duc &im_in, Imc2dsf &im_out );
```

Version française

Changement d’espace couleur de RVB vers YUV.

Author: Olivier Lezoray

proberts

Computes the Roberts gradient magnitude.

Synopsis

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

Description

proberts computes an approximation of the gradient magnitude of the input image *im_in*.

The algorithm uses the convolution with the following kernel:

```
|+0 -1|
|+1 +0|
```

The kernel is oriented in two directions: 0 and 90 degrees and the magnitude value is set to the maximum value between these two values.

The output image *im_out* is of the same type as the input image *im_in*.

Inputs

- *im_in*: a grayscale image.

Outputs

- *im_out*: an image of the same type as im_in.

Result

Returns SUCCESS or FAILURE.

Examples

Performs an edge detection for the tangram.pan image:

```
proberts tangram.pan b.pan
pbinarization 15 1e30 b.pan out.pan
```

See also

Edge detection

C++ prototype

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

Version française

Module du gradient de Roberts.

Author: Régis Clouard

protation

Performs rotation of image or region map.

Synopsis

protation *axis angle* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

protation rotates the input image *im_in* to build the output image *im_out*. The rotation is done along the specified *axis* with a specified angle.

The output image *im_out* keeps the same type than the input image *im_in*. The new introduced pixels are set to 0.

Notice: 4 successive rotations with 90° produces an output image slightly different from the original image due to calculus approximation.

Parameters

- *angle* is a real value measured in degree and it can be negative.
- *axis* is specified by an integer:
  - 0: rotation around the z axis (by default for 2D image).
  - 1: rotation around the y axis.
  - 2: rotation around the z axis.

Inputs

- *im_in*: an image or a region map.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Rotates tangram.pan to 90 degrees:

```
protation 0 90 tangram.pan a.pan
```

See also

Transformation

C++ prototype

```
Errc PRotation( const Img2duc &im_in; Img2duc &im_out; int axis,
float angle );
```

Version française

Construction de la rotation selon un axe du contenu d’une image.

Author: Régis Clouard

pround

Computes round integral value of image or graph.

Synopsis

pround *mode* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pround performs rounding of input values.

If *im_in* is an image, **pround** is applied on each pixel value:

```
pixel(im_out)=round(pixel(im_in))
```

The rounding type depends on the parameter value *mode*:

- 0: the nearest integer than the initial value (1.1=1; 1.9=2; -8.8=-9; -8.1=-8).
- 1: the nearest integer not greater than the initial value (1.1=1, 1.9=1; -8.8=-9; -8.1=-9);
- 2: the nearest integer not lower than the initial value (1.1=2; 1.9=2; -8.8=-8; -8.1=-8);

For color or multispectral image, rounding is computed separately on each band.

The output image *im_out* is of the same type as *im_in*.

If *im_in* is a graph rounding is applied on each node value.

Parameters

- *mode* specifies the type of rounding:
  - 0: the nearest integer than the initial value (1.1=1; 1.9=2; -8.8=-9; -8.1=-8).
  - 1: the nearest integer not greater than the initial value (1.1=1, 1.9=1; -8.8=-9; -8.1=-9);
  - 2: the nearest integer not lower than the initial value (1.1=2; 1.9=2; -8.8=-8; -8.1=-8);

Inputs

- *im_in*: a Float image or a graph.

Outputs

- *im_out*: a Float image or a graph.

Result

Returns SUCCESS or FAILURE.

Examples

```
plog a.pan b.panx
pround 0 b.pan c.pan
```

See also

Arithmetic

C++ prototype

```
Errc PRound( const Img2dsf &im_in, Img2dsf &im_out, int mode );
```

Version française

Arrondi d’une image de réels ou d’un graphe.

Author: Régis Clouard

pscrolling

Performs scrolling of image or region map.

Synopsis

pscrolling *direction shift* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pscrolling translates the content of the input image (or region map) *im_in* to *shift* pixels in the specified *direction*. Pixels that are moved out the image are set to the other side of the image.

Parameters

- *direction* is in the interval [0..2] where:
  - 0: along x axis.
  - 1: along y axis.
  - 2: along z xis.
- *shift* is a positive or negative integer. It is measured in pixel unit.

Inputs

- *im_in*: an image or a region map.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Scrolls the tangram.pan image of 50 pixels from the left:

```
pscrolling 0 50 tangram.pan a.pan
```

See also

ptranslate, Transformation

C++ prototype

```
Errc PScrolling( const Img3duc &im_in,Img3duc &im_out, int
direction, Long shift );
```

Version française

Construction de l’enroulé d’une image.

Author: Régis Clouard

psedesign

Designs structuring element as image.

Synopsis

psedesign *num_se halfsize* [*im_out*|-]

Description

psedesign allows to generate a structuring element as a new image.

The structuring element is designed from the type *num_se* and the size *halfsize*. For example, a *halfsize* of 1 for a *num_se*=3 gives a structuring element of type disc with size 3x3.

It is possible to create a structurant element with other shape from its specification in a text file (see ptxt2pan).

These structuring elements can then be used by the operators pseerosion and psedilatation.

Parameters

- The type of the structuring element is given by *num_se*:

  In 2D:

  - 0: diamond (4-connexity)
  - 1: square (8-connexity)
  - 2: disc
  - 3: horizontal line (-)
  - 4: vertical line (|)
  - 5: right oblique line (/).
  - 6: left oblique line (\).

  In 3D:

  - 10: bipyramid (6-connexity)
  - 11: cubic (26-connexity)
  - 12: sphere
  - 13: line in X (-)
  - 14: line in y (|)
  - 15: line in Z
- *size* gives the half-size of the structuring element. For example, a half-size of 1 for a square gives a structuring element of size 3x3.

Outputs

- *im_out*: a 2D or 3D image of bytes.

Result

Returns SUCCESS or FAILURE.

Examples

Performs geodesic black Top Hat with small 17x17 square structuring element:

```
psedesign 1 8 es.pan
pinverse tangram.pan i0.pan
psedilatation 1 i0.pan i1.pan
pdilatationreconstruction 8 i1.pan i0.pan i2.pan
pdif i2.pan out.pan
```

See also

Morphology, psedilatation, pseerosion, ptxt2pan

C++ prototype

```
Img2duc* PSEDesign( int numse, int halfsize );
```

Version française

Génération d’un élément structurant prédéfini.

Author: Régis Clouard

psedilatation

Performs morphological dilatation.

Synopsis

psedilatation *size* [*-m mask*] [*im_se*|-] [*im_in*|-] [*im_out*|-]

Description

psedilatation dilates the points of stronger contrast according to a structuring element.

Dilatation corresponds to the operation: replace the central pixel p by the maximum of its neighbors where the neighbors are specified by the structuring element.

```
dilatation(p) = MAX(neighbors(p)).
```

The structuring element is given in the input image *im_se*. It is a bytes image (Uchar) of the size of the structuring element. This image can be built from a textual file (see pxt2pan) or from the structuring element generator (see pedesign).

For a binary image, dilatation dilates white areas.

For the region maps, dilatation dilates only regions that touch the background and region with the higher label are privileged.

For the color images, the lexicographic order is used: initially by using band X, in the event of equality by using the band Y then band Z.

Parameters

- *size* specifies the number of iterations.

Inputs

- *im_in*: an image or a region map.
- *im_se*: an image of bytes.

Outputs

- *im_out*: an image (or a region map) of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Performs black Top Hat with small 17x17 square structuring element:

```
psedesign 1 8 es.pan
pinverse tangram.pan i0.pan
psedilatation 1 es.pan i0.pan i1.pan
perosionreconstruction 8 i1.pan i0.pan i2.pan
pdif i0.pan i2.pan out.pan
```

See also

Morphology, pdilatation, pseerosion

C++ prototype

```
Errc PSEDilatation( const Img2duc &im_in, const Img2duc &im_se,
Img2duc &im_out, int size );
```

Version française

Dilatation des points de fort contraste d’une image à partir d’un élément structurant donné.

Author: Régis Clouard

pseerosion

Performs morphological erosion.

Synopsis

pseerosion *size* [*-m mask*] [*im_se*|-] [*im_in*|-] [*im_out*|-]

Description

pseerosion erodes the points of stronger contrast according to a structuring element.

Erosion corresponds to the operation: replace the central pixel p by the minimum of its neighbors where the neighbors are specified by the structuring element.

```
erosion(p) = MIN(neighbors(p)).
```

The structuring element is given in the input image *im_se*. It is a bytes image (Uchar) of the size of the structuring element. This image can be built from a textual file (see ptxt2pan) or from the structuring element generator (see psedesign).

For a binary image, erosion erodes white areas.

For the region maps, erosion adds pixels with label=0 (background) at the points of erosion.

For the color images, the lexicographic order is used: initially by using band X, in the event of equality by using the band Y then band Z.

Parameters

- *size* specifies the number of iterations.

Inputs

- *im_in*: an image or a region map.
- *im_se*: an image of bytes (Uchar image).

Outputs

- *im_out*: an image (or a region map) of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Performs geodesic white Top Hat with small 17x17 square structuring element:

```
psedesign 1 8 es.pan
pseerosion 1 es.pan tangram.pan i1.pan
pdilatationreconstruction 8 i1.pan tangram.pan i2.pan
pdif tangram.pan i2.pan out.pan
```

See also

Morphology, psedilatation, perosion

C++ prototype

```
Errc PSEErosion( const Img2duc &im_in, const Img2duc &im_se, Img2duc
&im_out, int size );
```

Version française

Erosion des points de fort contraste d’une image à partir d’un élément structurant donné.

Author: Régis Clouard

psetborder

Sets image border to specified value.

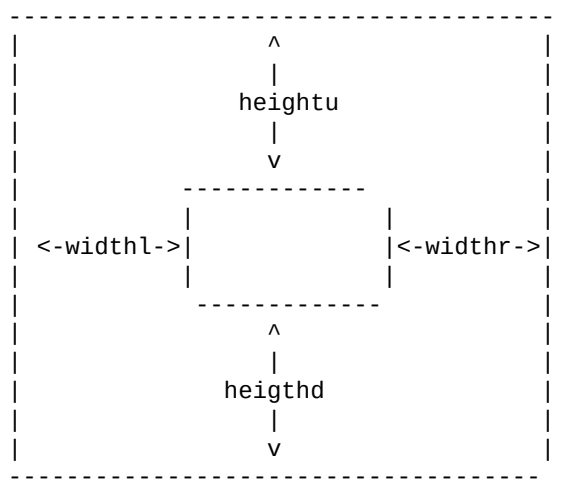
Synopsis

psetborder *widthl widthr heightu heightd depthf depthb val* [*im_in*|-] [*im_out*|-]

Description

psetborder sets the specified value *val* to the pixel on the border of the input image *im_in*. The border is defined by the dimensions *widthl*, *widthr*, *heightu*, *heightd*, *depthf*, *depthb* where *widthl* is the length of the left border, *widthr* is the length of the right border, *heightu* is the height of the upper border, *heightd* is the height of the lower border, *depthf* is the depth of the forward border and *depthf* is the depth of the backward border.

For a 2D image, the dimensions are:



For color and multispectral image, the value is set on each bands.

Parameters

- *depthf, depthb, heightu, heightd, widthl, widthr* specify the dimensions of the border.

  In case of 2D image, *depthf* and *depthb* are not used but must be given.

- *val* is the value set to each pixel of the border. The type depends on the input image type.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image with the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Selects objects of a tangram.pan subimage (in.pan) that do not touch the border of the image reg.pan.

```
pextractsubimage  30 30 0 150 150 0 tangram.pan in.pan
psetcst 0 ii.pan i1.pan
psetborder 1 1 1 1 1 1 255 i1.pan i2.pan
pdilatationreconstruction 8 i2.pan ii.pan i3.pan
pdif in.pan i3.pan out.pan
```

See also

Utility

C++ prototype

```
Errc PSetBorder( const Img3duc &im_in, Img3duc &im_out, Long depthf,
Long depthb, Long heightu, Long heigthd, Long widthl, Long widthr,
Uchar val );
```

Version française

Affectation d’une valeur sur le bord d’une image.

Author: Régis Clouard

psetcst

Sets constant to image, graph or region map.

Synopsis

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

Description

psetcst builds the new output *im_out* by replacing each value by the specified constant value.

If *im_in* is an image, **psetcst** sets the specified value to each pixel. This operator can be used to create a new image *im_out* from the properties of the input image *im_in*.

For color or multispectral image, **psetcst** is computed separately on each band.

For region map, **psetcst** sets the specified value to each label.

For graph, **psetcst** sets the specified value to each node value.

The output file is of the same type as the input file.

Parameters

- *cst* is a real value. If needed the constant is casted to match the type of im_in data.

Inputs

- *im_in*: an image, a graph or a region map.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

For region map, returns the new higher label value (ie. the constant value).

Examples

Creates a.pan with the same property as tangram.pan and sets all its pixels to 10.

```
psetscst 10 tangram.pan a.pan
```

See also

Arithmetic

C++ prototype

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

Version française

Affectation d’une valeur à une image, un graphe ou une carte de région.

Author: Régis Clouard

psetslice

Sets slice to 3D image or region map.

Synopsis

psetslice *slice* [*-m mask*] [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

psetslice builds a new 3D image *im_out* from the input 3D image *im_in1* where the slice number *slice* is replaced by the given 2D image *im_in2*.

The result image *im_out* is of the same type as the two input images.

Parameters

- *slice* specifies the number of the slice to be replaced. It is an integer between 0 and the total number of *im_in1* slice minus 1.

Inputs

- *im_in1*: a 3D image or 3D region map.
- *im_in2*: a 2D image or 2D region map.

Outputs

- *im_out*: a 3D image or a 3D region map.

Result

Returns SUCCESS or FAILURE.

Examples

Replaces slice #10 (11th slice) of the a3d.pan with the 2D image a2d.pan:

```
psetslice 10 a3d.pan a2d.pan b3d.pan
```

See also

Utility, pgetslice, premoveslice

C++ prototype

```
Errc PSetSlice( const Img3duc &im_in1, const Img2duc &im_in2,
Img3duc &im_out, long slice );
```

Version française

Remplacement d’un plan d’une image 3D par une image 2D.

Author: Régis Clouard

psetstatus

Sets the current status value.

Synopsis

psetstatus *value*

Description

psetstatus sets the current status value. The status value is the value returned after each execution of a Pandore operator. This value is operator dependent and can be of the following types:

- an integer;
- a real;
- a character;
- a string;
- SUCCESS / FAILURE.

The status value can be printed out by operator pstatus.

setstatus is useful to build new operator as pure shell script since it allows to set the result of the operator. For example, the following shell script becomes a Pandore operator that checks if an image exist and sets the status to SUCCESS or FAILURE from the result:

```
#!/bin/sh
if [ -f $1 ]
then
psetstatus SUCCESS
else
psetstatus FAILURE
fi
```

Parameters

- *value* can be:
  - an integer;
  - a real;
  - a character;
  - a string;
  - or SUCCESS / FAILURE.

Result

The value set.

Examples

Sets the current value to SUCCESS.

```
psetstatus SUCCESS
pstatus {-> returns value SUCCESS }
```

Sets the current value to -12e20.

```
psetstatus -12e20
pstatus {-> returns value -12e20 }
```

See also

Information, pstatus

C++ prototype

Not a function.

Version française

Affecte la valeur du statut courant à une valeur donnée.

Author: Régis Clouard

psetsubband

Sets subband into DWT image.

Synopsis

psetsubband *scale subband* [*im_in1*|-] [*im_in2*|-] [*im_out*|-]

Description

psetsubvband inserts a subband to a DWT image at the specified *scale*. The subband *im_in2* is an image and it is added to the input DWT image *im_in1*. At each scale, image are numbered as follows:

```
[1][2]
[3][4]
```

- 1: subband LL of approximate coefficients.
- 2: subband LH of detail coefficients.
- 3: subband HL of detail coefficients.
- 4: subband HH of detail coefficients.

Parameters

- *scale* specifies the scale analysis of the DWT image.
- *subband* specifies the number of the subband [1..4] to be inserted at the specified scale.

Inputs

- *im_in1*: a 2D grayscale image of DWT.
- *im_in2*: a 2D image with the convenient size.

Inputs

- *im_out*: an image of the same type as the input image *im_in1*.

Result

Returns SUCCESS or FAILURE.

Examples

Threshold the LL of the DWT analysis of a square:

```
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
```

See also

Frequency

C++ prototype

```
Errc PSetSubband( const Img2dsf &im_in1, const Img2dsf &im_in2,
Img2dsf &im_out, int scale, int subband);
```

Version française

Insertion d’une sous-bande dans une image de DWT.

Author: Ludovic Soltys

pshapedesign

Creates a new image with synthetic shape.

Synopsis

pshapedesign *w h d num diameter length* [*im_out*|-]

Description

pshapedesign creates a new image from the specified dimensions (*w,h,d*) with a synthetic shape. The parameter *num* determines the shape type. The background is set to 0 and the shape is set to 255. Shapes are oriented vertically.

The center of the shape is center in the image. Height, width and depth of the shape are deduced from the *diameter* and the *length* parameters.

Parameters

- *w,h,d* specify the dimension of the new image, respectively the width, the height, and the depth. If d=0 then the output image is a 2D image, and if h=0 then the output image is a 1D image.
- *num* determines the shape type:

  2D
  - 0- empty
  - 1- disc
  - 2- square
  - 3- rectangle

  3D
  - 10- empty
  - 11- sphere
  - 12- cube
  - 13- parallelepiped
  - 14- cylinder
  - 15- bipyramid
- *diameter* specifies the diameter of a symmetrical shape, or the height and depth of an asymmetrical shape.
- *length* specifies the length of asymmetrical shape. For symmetrical shape, *length* is ignored.

Outputs

- *im_out*: a gray level image of Uchar (Img2duc or Img3duc).

Result

Returns SUCCESS or FAILURE.

Examples

Builds a synthetic image (a square) to illustrate the Gibbs phenomenon in wavelets analysis.

```
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
```

See also

Utility

C++ prototype

```
Errc PShapeDesign( const Img2duc &im_out, long num, int diameter,
int length );
```

Version française

Création d’une image vierge ou avec une forme synthétique prédéfinie.

Author: Jean-Marie Janik

psharp

Performs contrast sharpening using laplacian unsharp masking.

Synopsis

psharp *connexity degree* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

psharp performs a contrast sharpening of the input image *im_in* using the Laplacian unsharp masking. The objective of sharpening is to highlight fine details and to enhance details that are blurred. It consists in shrinking the width of intensity variation without affecting the mean intensity of regions on both sides of the variation.

The unsharp masking algorithm is based on the subtraction of a *degree* times of the input image with the laplacian of the image. It is implemented by a spatial filtering with the following filters that depends on the *connexity*.
For example, 2D filters are:

```
      4-connexity                 8-connexity
   0       -1       0           -1      -1      -1
  -1    4*degree   -1    or     -1    8*degree  -1
   0       -1       0           -1      -1      -1
```

For 3D filters, the center is set with 6*degree in case of 6-connexity, or 26*degree in case of 26-connexity.

Parameters

- *connexity* specifies the type of connexity between neighbours (4 and 8 for 2D image and 6 or 26 for 3D image).
- *degree* is a real value that specifies the degree of sharpening. The more is the degree the less is the effect of sharpening. Typical values are 0.7, 1.0, 1.7, 2, 7.

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image with the same properties as im_in.

Result

Returns SUCCESS or FAILURE in case of invalid parameter values.

Examples

Sharpens the tangram.pan using the laplacian (which corresponds to degree=1).

```
psharp 8 1 tangram.pan a.pan
```

See also

Lut transform

C++ prototype

```
Errc PSharp( const Img2duc &im_in, Img2duc &im_out, int connexity,
float degree );
```

Version française

Rehaussement du contraste par convolution.

Author: Régis Clouard

pshen

Computes gradient magnitude and maxima localization using Shen-Castan's algorithm.

Synopsis

pshen *sigma* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pshen computes the gradient magnitude and performs the maxima localization. The output image can then be used to locate the contours. The output image *im_out* is built with the maximum magnitude value in the direction of the gradient. Other values are set to 0.

The gradient magnitude value reflects the amount of grayscale variation in this point. The more is the variation the greater is the value.

The gradient extraction and localization is done in three steps :

1. smoothing ;
2. gradient computing ;
3. local maxima extraction.

Note: The image border of size 1 is set to 0.

Parameters

- *sigma* specifies the strength of the smoothing The lower is value, the stronger is the smoothing thus less there are contour points in the output image. A typical value is 1.

Inputs

- *im_in*: a 2D grayscale image.

Outputs

- *im_out*: an image of the same type as im_in.

Résultat

Returns SUCCESS or FAILURE.

See also

Edge detection

Examples

Performs an edge detection for the tangram.pan image:

```
pshen 1 tangram.pan a.pan
pbinarization 10 1e30 a.pan b.pan
```

C++ prototype

```
Errc PShen( const Img2duc &im_in, Img2duc &im_out, float sigma );
```

Version française

Détection et localisation des contours de Shen-Castan.

Author: Carlotti & Joguet

pshensmoothing

Performs Shen-Castan smoothing on image.

Synopsis

pshensmoothing *sigma* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pshensmoothing performs Shen-Castan smoothing of the input image *im_in*.

Parameters

- *sigma* specifies the strength of the smoothing The lower is value, the stronger is the smoothing. A typical value is 1.5.

Inputs

- *im_in*: a 2D image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Performs Shen smoothing on tangram.pan:

```
pshensmoothing 1 tangram.pan out.pan
```

See also

Filtering

C++ prototype

```
Errc PShenSmoothing( const Img2duc &im_in, Img2duc &im_out, float
sigma );
```

Version française

Lissage de Shen&Castan.

Author: Carlotti & Joguet

psig

Builds the Sphere of Influence Graph.

Synopsis

psig [*-m mask*] [*gr_in*|-] [*gr_out*|-]

Description

psig builds the Sphere of Influence Graph. Edges of the input graph *gr_in* are cut when the related nodes are not in the same sphere of influence. Spheres of influence are circle centered on the nodes and the radius is the distance between two nodes. If the circles of two neighbor nodes intersect, then the edge between the two nodes is kept otherwise it is cut.

The values of each node is set to 1.

The distance between two nodes uses the euclidean distance of their related coordinates.

Inputs

- *gr_in*: a graph.

Outputs

- *gr_out*: a graph.

Result

Returns SUCCESS or FAILURE.

Examples

```
psig g1.pan g2.pan
```

See also

Graph

C++ prototype

```
Errc PSig( const Graph &gr_in, Graph &gr_out );
```

Version française

Construction de la sphère d’influence d’un graphe.

Author: François Angot

psigmafiltering

Performs sigma filtering on image.

Synopsis

psigmafiltering *halfsize eps nbmin* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

psigmafiltering applies the sigma filter algorithm on the input image *im_in*.

Each pixel is replaced by the mean value of some of its (halfsize*2+1) neighbors. Only neighbors that have a value close to the central pixel are considered for the mean. Close means which difference is lower than the specified parameter value *eps*. If the number of used neighbors is lower than the specified parameter value *nbmin*, than the central pixel is replaced par the mean of all its neighbors.

The image border (of size halfsize) is not considered for processing. The output image border is just a copy of the input image border.

Parameters

- *halfsize* specifies the size of the neighborhood. For example, if halfsize=1 then the neighborhood is 3x3 around the central pixel.
- *eps* specifies the maximum difference allowed from the central pixel. It is a grayscale value.
- *nbmin* specified the minimum number of neighbors that can be used to computed the new central pixel value. It is a positive integer.

Inputs

- *im_in*: a grayscale image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Applies a sigma filter to tangram.pan:

```
psigmafiltering 1 20 2 tangram.pan out.pan
```

See also

Filtering

C++ prototype

```
Errc PSigmaFiltering( const Img2duc &im_in, Img2duc &im_out, Short
halfsize, Ushort eps, Ushort nbmin );
```

Version française

Lissage par filtre adaptatif basé sur le choix des voisins.

Author: Régis Clouard

psizeselection

Selects regions from size measures.

Synopsis

psizeselection *relation width height depth* [*-m mask*] [*rg_in*|*-*] [*rg_out*|*-*]

Description

psizeselection selects regions from their size measures (depth, height, and width). The parameter *relation* specifies the order relation on the size values that are used to select or not a region.

If one of size parameter =-1 then this size feature is not taken into account to select regions.

Parameters

- *relation* is an integer from [-2,2] which specifies the order relation on the size value:
  - *relation* = 2: regions >= size
  - *relation* = 1: regions > size.
  - *relation* = 0: regions = size.
  - *relation* = -1: regions < size.
  - *relation* = -2: regions <= size.
- *depth* is an integer defined in pixel unit. If the parameter value =-1 then this size is not taken into account.
- *height* is an integer defined in pixel unit. If the parameter value =-1 then this size is not taken into account.
- *width* is an integer defined in pixel unit. If the parameter value =-1 then this size is not taken into account.

Inputs

- *rg_in*: a region map.

Outputs

- *rg_out*: a region map.

Result

Returns the number of selected regions.

Examples

Selects all regions with width = 50 pixels in the input region map rin.pan:

```
psizeselection 0 50 -1 0 rin.pan rout.pan
```

See also

Region

C++ prototype

```
Errc PSizeSelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, long depth, long height, long width );
```

Version française

Sélection de régions sur leur taille.

Author: Régis Clouard

pskeletonization

Computes the skeleton of 2D objects.

Synopsis

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

Description

pskeletonization computes the skeleton of the binary objects in the 2D image *im_in*. Binary objects are regions with connected non null pixels.

The algorithm rests on a succession of thinning until obtaining a stable structure not being able to be thinned, ie. whose elements are lines thickness 1 pixel.
Thinning is obtained by 8 masks of erosion in 8 possible directions N, NW, W, SW, S, SE, E, NE. A mask indicates the possible shape of a line according to the selected direction. For example the mask EAST is as follows:

```
if there is such a configuration around the central pixel ( x can be 0 or 1):
   0  x  1
   0  1  1
   0  x  1
then the central pixel is set to 1   else it is set to 0.
```

Inputs

- *im_in*: a 2D binary image.

Outputs

- *im_out*: a 2D binary image.

Result

Returns SUCCESS or FAILURE.

Examples

Creates the skeleton of the tangram pieces:

```
pbinarization 95 1e30 tangram.pan i1.pan
pskeletonization i1.pan out.pan
```

See also

Morphology, phomotopicskeletonization

C++ prototype

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

Version française

Squelettisation d’objets binaires.

Author: Régis Clouard

psnnfiltering

Performs Symmetric Nearest Neighborhood filtering on image.

Synopsis

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

Description

psnnfiltering applies a Symmetric Nearest Neighborhood filter to the input image *im_in*. Each pixel of the input image is replaced by the mean of its symmetric nearest neighbors.

The image border (of size 1) is not considered for processing. The output image border is just a copy of the input image border.

Inputs

- *im_in*: a grayscale image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Applies the Symmetric Nearest Neighborhood filter to tangram.pan image:

```
psnnfiltering tangram.pan out.pan
```

See also

Filtering

C++ prototype

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

Version française

Lissage par filtre adaptatif : le plus proche voisinage symetrique.

Author: Abderrahim Elmoataz

psnr

Computes the Signal-to-Noise Ratio.

Synopsis

psnr [*im_in1*|-] [*im_in2*|-]

Description

psnr measures the ratio between the meaningful information and the noise in an image. It is computed from the two input images: *im_in1* is the initial image that contains the signal and noise and *im_in2* is the restored or enhanced version of *im1_in* that is supposed to contain only meaningful information. Consequently, the higher is the SNR, the better is the signal and better is the related image processing (restoration or enhancement).

Because many signals have a very wide dynamic range, SNR is expressed in terms of decibel (dB).

SNR is defined as follows:

$$S = 10 * \log_{10}(R_{12})$$
$$R_{12} = \text{sum}\{(ims1)^2\} / \text{sum}(ims2-ims1)^2\}$$

Input images *im_in1* and *im_in2* must have the same dimensions and the same type.

For color or multispectral images, the definition of SNR is the same except that each sum uses all bands.

Note: R_{12} is dependent not only on the difference *ims1-ims2*, but also on *ims1*. Thus, the signal-to-noise ratio value is dependent to input images and is often used for processing comparison with the same input images.

Inputs

- *im_in1*: an image.
- *im_in2*: an image (restored or enhanced version of im_in1).

Example

Computes the SNR after a meanfilter smoothing:

```
pmeanfiltering 2 tangram.pan i1.pan
psnr tangram.pan i1.pan
pstatus
```

Result

Returns the ratio value as a positive real value expressed in dB.
(Use `pstatus` to get this value).

See also

Evaluation, pmse, ppsnr

C++ prototype

```
Errc PSNR( const Img2duc &im_in1, const Img2duc &im_in2 );
```

Version française

Calcul du rapport signal sur bruit.

Author: Régis Clouard

psobel

Computes the Sobel gradient magnitude.

Synopsis

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

Description

psobel computes an approximation of the gradient magnitude of the input image *im_in*.

The algorithm uses the convolution with the following kernel:

```
|+1 +2 +1|
|+0 +0 +0|
|-1 -2 -1|
```

The kernel is oriented in four directions: 0, 45, 90, 135 degrees and the magnitude value is set to the maximum value between these four values.

The output image *im_out* is of the same type as the input image *im_in*.

Inputs

- *im_in*: a grayscale image.

Outputs

- *im_in*: an image of the same type as im_in.

Result

Returns SUCCESS or FAILURE.

Examples

Performs an edge detection for the tangram.pan image:

```
psobel tangram.pan b.pan
pbinarization 45 1e30 b.pan out.pan
```

See also

Edge detection

C++ prototype

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

Version française

Module du gradient de Sobel.

Author: Régis Clouard

psphericityselection

Selects regions from sphericity degree.

Synopsis

psphericityselection *relation threshold* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

psphericityselection selects regions from their sphericity degree. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

Sphericity measures the degree to which an object approaches the shape of a sphere. It is calculated using:

```
sphericity = rayon_inscribing / rayon_circumscribing.
```

The maximum value is 1.0 for a circle.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is a real value from [0..1] which specifies a sphericity degree.

Inputs

- *rg_in*: a 2D region map.

Outputs

- *rg_out*: a 2D region map.

Result

Returns the number of selected regions.

Examples

Selects regions with the highest sphericity degree:

```
psphericityselection 3 0 a.pan b.pan
```

See also

Region

C++ prototype

```
Errc PSphericitySelection( const Reg2d &rg_in, Reg2d &rg_out, int
relation, float threshold );
```

Version française

Sélection de régions sur leur valeur de sphéricité.

Author: Régis Clouard

psplitimage

Splits image into 4 subimages.

Synopsis

psplitimage [*im_in*|-] [*im_out1*|-] [*im_out2*|-] [*im_out3*|-] [*im_out4*|-]

Description

psplitimage splits the input image *im-in* into four subimages *im_out1*, *im_out2*, *im_out3* and *im_out4* with the same size:

```
[im_out1][im_out2]
[im_out3][im_out4]
```

If the *im_in* size is odd, then the first rectangle is smaller than the second.

Inputs

- *im_in*: a 2D image.

Outputs

- *im_out1*, *im_out2*, *im_out3*, *im_out4*: 2D images.

Result

Returns SUCCESS or FAILURE.

Examples

Builds a synthetic image to illustrate the Gibbs phenomenon in wavelets analysis.

```
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
```

See also

Utility, pmergeimages

C++ prototype

```
Errc PSplitImage( const Img2dsf &im_in, Img2dsf &im_out1, Img2dsf
&im_out2, Img2dsf &im_out3, Img2dsf &im_out4 );
```

Version française

Eclatement d’une image en 4 sous-images.

Author: Ludovic Soltys

psqrt

Computes square root of image or graph.

Synopsis

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

Description

psqrt computes the square root of the input *im_in*.

If *im_in* is an image then the new image *im_out* is built with the square root of each pixel:

```
pixel(im_out)=sqrt(pixel(im_in))
```

The output image *im_out* is always a Float image.

For color or multispectral image, the square root is computed separately on each band.

If *im_in* is a graph then the new graph *im_out* is built with the square root of each node value.

Inputs

- *im_in*: an image or a graph.

Outputs

- *im_out*: a Float image or a graph.

Result

Returns SUCCESS or FAILURE.

Examples

```
psqrt tangram.pan a.pan
```

See also

Arithmetic

C++ prototype

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

Version française

Racine carrée d’une image ou d’un graphe.

Author: Régis Clouard

pstatus

Prints out the value returned by the last executed operator.

Synopsis

`pstatus`

Description

pstatus displays the value returned by the last execution of Pandore operator.

Each Pandore operator returns a value after its execution. This value is operator dependent and can be of the following types:

- an integer;
- a real;
- a character;
- a string
- SUCCESS / FAILURE.

Such a value can then be used to test the completeness of an operator execution or be used as parameter value for an another operator.

- For example, on UNIX/LINUX/MACOS system:

```
plabeling 8 a.pan b.pan
a=‘pstatus‘
echo "Number of regions=%a%"
```

- For example, on MsDos it is necessary to use the ’pset’ command to set a variable with the status value:

```
plabeling 8 a.pan b.pan
call pstatus
call pset a
echo "Number of regions=%a%"
```

Result

No result value.

Examples

Unix/Linux/MACOS: Displays the number of regions which resulted in a segmentation process based on interclass variance maximization:

```
pvariancebinarization tangram.pan a.pan
plabeling 8 a.pan b.pan
a=‘pstatus‘
echo "Number of regions=%a%"
```

MsDos: Displays the number of regions which resulted in a segmentation process based on interclass variance maximization:

```
pvariancebinarization tangram.pan a.pan
plabeling 8 a.pan b.pan
call pstatus
call pset a
echo "Number of regions=%a%"
```

See also

Information, psetstatus

Version française

Affichage de la valeur retournée par la dernière exécution d’un opérateur Pandore.

Author: Régis Clouard

pstereogram

Builds Autostereogram image.

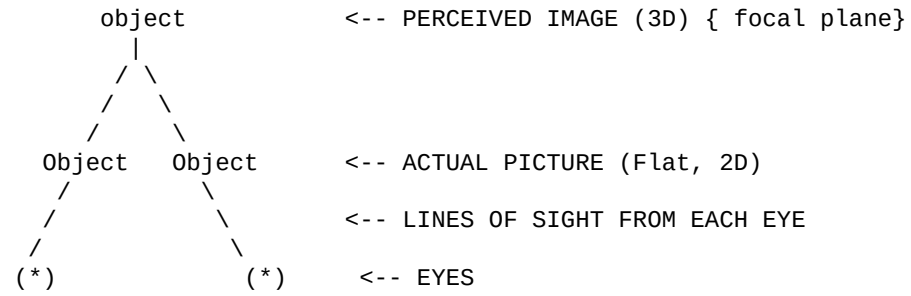
Synopsis

pstereogram [*im_in1*|-] [*im_im2*|-] [*im_out*|-]

Description

pstereogram builds a stereogram image form a depth map *im_in1* and a pattern image *im_in2*.

A stereogram is an image in which stereoscopic information are encoded. To look at a stereogram, one has to consider that the focal plane is behind the image.



The depth map describes how deep each part of the scene is. It is represented as a gray scale image, where each pixel encodes the depth of the related point in the scene. Values belongs to [0..255] where 0 is the minimum distance and 255 the maximum distance. The depth image can be built from the projection of a 3D image to 2D image.

The pattern image is used as a tile image. It must be at least as wide as the maximum stereo separation otherwise there is a problem (for example 64x64).

Inputs

- *im_in1*: a 2D gray level image (Img2duc). Each pixel encodes the distance from the observer of the related point in the scene.
- *im_in2*: a 2D color image (Imc2duc).

Outputs

- *im_out*: a 2D color image (Imc2duc).

Result

Returns SUCCESS or FAILURE.

Examples

Builds c.pan from the depth image a.pan and the tile image b.pan. a.pan is builds from the 3D image a3d.pan:

```
pgraylevel2depth 50 a3d.pan a.pan
pstereogram a.pan b.pan c.pan
```

See also

Miscellaneous, pstereogram

C++ prototype

```
Errc PStereogram( const Img2duc &imp, const Imc2duc &imt, Imc2duc &im_out );
```

Version française

Construction d’une image stéréogramme couleur.

Author: Régis Clouard

psub

Performs subtraction between images or graphs and non symmetrical difference between region maps.

Synopsis

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

Description

psub computes the difference between the two inputs *im_in1* and *im_in2*.

If *im_in1* and *im_in2* are images then the new image *im_out* is built with the difference between each pixel:

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

The two inputs must be of the same type.

The output type *im_out* depends on input type:

- Long if inputs are Uchar images.
- Long if inputs are Long images.
- Float if inputs are Float images.

For color or multispectral image, the difference is computed separately on each band.

If *im_in1* and *im_in2* are graphs then the new graph *im_out* is built with the difference between each node values.

If *im_in1* and *im_in2* are region maps **pdif** computes the symmetrical difference between region maps.

Inputs

- *im_in1*: an image, a graph or a region map.
- *im_in2*: an image, a graph or a region map.

Outputs

- *im_out*: an image, a graph or a region map.

Result

Returns SUCCESS or FAILURE.

For region map, returns the new higher label value.

Examples

```
psub a.pan b.pan c.pan
```

See also

Arithmetic

C++ prototype

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

Version française

Soustraction d’images ou de graphes et différence non symétrique entre cartes de régions.

Author: Régis Clouard

psubsampling

Sub-sampling of an image.

Synopsis

psubsampling *factor* [*im_in*|-] [*im_out*|-]

Description

psubsampling builds a new image with a subsample version of the input image.

Subsampling consists in building a tile of size *factor*xfactor in the output image with the mean color value of the corresponding tile in the input image.

Parameters

- *factor* is a positive integer.

Inputs

- *im_in*: an image, a region map, or a graph.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

Sub-sampling of a factor 2:

```
psubsampling 2 tangram.pan a.pan
```

See also

Miscellaneous.

C++ prototype

```
Errc Psubsampling( const Img2duc &im_in, Img2duc &im_out, int factor
);
```

Version française

Sous-échantillonnage d’une image

Author: Régis Clouard

psubval

Subtracts constants stored in collection to image bands.

Synopsis

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

Description

psubval builds the new output *im_out* by subtracting constants stored in the collection *col_in* to each band of the input image *im_in*. The first constant in the collection is subtracted to the first band, the second constant to the second, etc.

The values are clipped if they are greater than the maximum allowed value or lower than the minimum:

```
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;
```

The output file is of the same type as the input file.

Inputs

- *col_in*: a collection with a number of float values equals to the number of bands of the input image.
- *im_in*: an image.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Subtracts the mean value of tangram.pan image to tangram.pan:

```
pmeanvalue tangram.pan col.pan
psubval col.pan tangram.pan a.pan
```

More examples

See also

Arithmetic

C++ prototype

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

Version française

Soustraction d’une image avec des constantes stockÃ©es dans une collection.

Author: Régis Clouard

psumvalue

Measures global sum of grayscale image or graph.

Synopsis

psumvalue [*im_in*|-] [*col_out*|-]

Description

psumvalue calculates the total sum of the pixels of the input image *im_in* or of the nodes of the input graph.

For the graph, the sum is calculated from the node values.

The sum values for each band are stored in the collection *col_out*.

Notice : This operator is not maskable.

Inputs

- *im_in*: an image or a graph.

Result

Returns the sum value (for the first band only). This value can be get using operator **pstatus**.

Examples

Measures the global sum of the tangram.pan (Unix version):

```
psumvalue tangram.pan col.pan
var=‘pstatus‘
echo "Sum = $val"
```

Measures the global sum of the tangram.pan (MsDos version):

```
psumvalue tangram.pan col.pan
call pstatus
call pset var
echo Sum = %val%
```

See also

Image Features Extraction

C++ prototype

```
Float PSumValue( const Img2duc &im_in, Collection & col_out );
```

Version française

Calcul de la somme des valeurs de pixels (ou de sommets).

Author: Régis Clouard

psuperimposition

Mask surimposition onto image.

Synopsis

psuperimposition *color_mask* [*im_in*|-] [*mk_in*|-] [*im_out*|-]

Description

psuperimposition builds the output image *im_out* by adding the mask pixels in the specified band of the input image. For example, if color_mask=1 then the mask is adding in the red band.

This operator is useful to visualize contours extracted from an image directly on this image.

The output image is of the same type as the input image.

Parameters

- *color_mask* specified the colorization mask. It acts like a bit mask. For example, if we consider a color image with 3 bands:
  - mask=0: the mask is painted in black;
  - mask=1: the mask is painted in red;
  - mask=2: the mask is painted in green;
  - mask=3: the mask is painted in red+green:yellow;
  - mask=4: the mask is painted in blue;
  - mask=5: the mask is painted in red+blue:purple;
  - mask=5: the mask is painted in green+blue:magenta;
  - mask=7: the mask is painted in white;

Inputs

- *im_in*: an image.
- *mk_in*: a gray image.

Outputs

- *im_out*: an image of the same type as the input image.

Examples

Visualize the contours extracted from the tangram image in the trangram image

```
pgradient 1 tangram.pan i1.pan i2.pan
pbinarization 30 1e30 i1.pan i3.pan
pbinarization 60 1e30 i1.pan i4.pan
pgeodesicdilatation 1 1 -1 i4.pan i3.pan i4.pan
psuperimposition 1 tangram.pan i4.pan out.pan
```

Result

Returns SUCCESS or FAILURE.

See also

Visualization

C++ prototype

```
Errc PSuperimposition( const Imc2duc &im_in, const Img2duc
&mk_in, Imc2duc &im_out, int color_mask);
```

Version française

Surimposition d’un masque sur une image.

Author: RÃ©gis Clouard

psusan

Performs Susan corner detection.

Synopsis

psusan *threshold* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

psusan is a corner detector. Corners are T, L or Y junctions or points with strong texture variation. Corners correspond to double discontinuities of the intensity function caused by discontinuities in the reflectance or the depth functions.

The susan algorithm consists in:

1. Center a circular mask of radius 3 on each pixel.
2. Compute the number of pixels in the mask with the same value than the center.
3. Threshold to produce the image with the strength of each pixel.
4. Suppress all non maximum pixels to keep only corners.

The output image *im_out* is a Long image that encodes for each pixel the strength of the response at this point.

Parameters

- *threshold* determines the maximum contrast between two pixels in the same region. The higher is the threshold, the less there are corners in the result. A typical value is 20.

Inputs

- *im_in*: a 2D image.

Outputs

- *im_out*: a Long image.

Result

Returns SUCCESS or FAILURE.

Examples

Extracts corners from image tangram.pan and superimposes corners on the initial image.

```
psusan 20 tangram.pan a.pan
pbinarization 1000 1e30 a.pan b.pan
padd b.pan tangram.pan out.pan
```

See also

Points of interest

C++ prototype

```
Errc PSusan( const Img2duc &im_in, Img2dsl &im_out, int threshold );
```

Version française

Détection de points d’intérêt selon l’algorithme SUSAN.

Author: Régis Clouard

pthresholding

Performs thresholding on image, region map or graph.

Synopsis

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

Description

pthresholding builds the output image *im_out* with the pixels of the input image *im_in* that have a value greater or equal than *low* or lower or equal than *high*. Other values are set to 0:

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

If *high* is lower than *low* then **pthresholding** performs an inverse thresholding:

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

For region map, **pthresholding** selects region with a label value greater or equal than *low* or lower or equal than *high*. There is no relabeling, output regions keep the same label than the input region.

For graph, **pthresholding** operates on graph nodes.

Parameters

- *low* and *high* specify the gray level bounds. Values are from the gray levels of the input image (eg. [0..255] for byte image, [-2147483648..+2147483648] for long integer image).

  Tip: If *high* is lower than *low* than **pthresholding** performs an inverse thresholding.

  Tip: If *high* is greater than the maximum gray level then high is set with the maximum value (respectively for *low*).

Inputs

- *im_in*: an image, a region map or a graph.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE in case of bad parameters.

Examples

Selects pixels of the tangram pieces:

```
pthresholding 100 1e30 tangram.pan out.pan
```

See also

Thresholding

C++ prototype

```
Errc PThresholding( const Img2duc &im_in, Img2duc &im_out, Uchar
low, Uchar high );
```

Version française

Seuillage d’une image selon la valeur de pixel.

Author: Régis Clouard

ptiff2pan

Converts TIFF image file to Pandore image file.

Synopsis

ptiff2pan *im_in* [*im_out*|-]

Description

ptiff2pan converts a TIFF image file to a Pandore image file. The result Pandore image type *im_out* depends on the input TIFF image type *im_in*.

Note: Only uncompressed TIFF image can be converted. It might be necessary to use another image converter to convert compressed TIFF file to uncompressed TIFF file.

Inputs

- *im_in*: a TIFF image file.

Outputs

- *im_out*: a 2D Pandore image file.

Result

Returns SUCCESS or FAILURE.

Examples

```
ptiff2pan image.tiff a.pan
```

See also

Conversion, ppan2tiff

C++ prototype

```
Img2duc* PTiff2Pan( const char *filename );
```

Version française

Conversion d’une image TIFF en une image Pandore.

Author: Régis Clouard

ptranslation

Performs translation of image or region map.

Synopsis

ptranslation *direction shift* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

ptranslation translates the content of the input image (or region map) *im_in* to *shift* pixels in the specified *direction*.

The new added pixel values are set to 0.

Parameters

- *direction* is in the interval [0..2] where:
  - 0: along x axis.
  - 1: along y axis.
  - 2: along z xis.
- *shift* is a positive or negative integer. It is measured in pixel unit.

Inputs

- *im_in*: an image or a region map.

Outputs

- *im_out*: an object of the same type as *im_in*.

Result

Returns SUCCESS or FAILURE.

Examples

Translates the tangram.pan image of 50 pixels from the left:

```
ptranslation 0 50 tangram.pan a.pan
```

See also

pscrolling, Transformation

C++ prototype

```
Errc PTranslation( const Img3duc &im_in,Img3duc &im_out, int
direction, Long shift );
```

Version française

Construction du translaté d’une image.

Author: Régis Clouard

ptransposition

Performs a transposition of image.

Synopsis

ptransposition *direction* [*im_in*|-] [*im_out*|-]

Description

ptransposition builds a new image from the transpose transformation of the input image *im_in* according to a specified axis. Transposing an image array interchanges array dimensions, reflecting an image about a diagonal. The transpose function reverses the order of the dimensions.

Parameters

- *direction* indicates the transposition direction:
  - 0 = transposition where x becomes y and y becomes x.
  - 1 = transposition where x becomes z and z becomes x.
  - 2 = transposition where y becomes z and z becomes y.

  For 2D image, this parameter is ignored since it corresponds to direction=0.

Inputs

- *im_in*: an image.

Outputs

- *im_in*: an image of the same type as *im_in*.

Examples

Transposes the tangram.pan image:

```
ptransposition 0 tangram.pan a.pan
```

Result

Returns SUCCESS ou FAILURE.

See also

Transformation

C++ prototype

```
Errc PTransposition( const Img2duc &im_in, Img2duc &im_out, int
direction );
```

Version française

Construit le transposé d’une image selon un axe.

Author: Régis Clouard

ptrianglerescale

Performs a rescaling of image using triangular filter.

Synopsis

ptrianglerescale *zoomx zoomy zoomyz* [*im_in*|-] [*im_out*|-]

Description

ptrianglerescale uses a convolution kernel to interpolate the pixel of the input image *im_in* in order to calculate the pixel value of the output image *im_out*. The interpolation consists in weigthing the input pixels influence on the output pixels. The weights are relative to the position of the output pixels and are given by the triangular filter:

```
T(x) = | 1-|x| if -1 < x < 1
       | 0 sinon
```

For example, if the image is scaled by 3, then each output pixel is:

```
for i in [-2, 2]
  for j in [-2, 2]
      im_out[p] += T(i)*T(j)*im_in[p]
```

To rescale region map or graph, use the operator prescale.

Parameters

- *zoomx*, *zoomy*, *zoomz* are positive real values.
  - if a zoom factor is > 1 then the image is enlarged along the related axis.
  - if a zoom factor is < 1 then the image is shrunk along the related axis.

  (*zoomz* is ignored for 2D images but must be given).

Inputs

- *im_in*: an image.

Outputs

- *im_out*: an image of the same type as the input image.

Result

Returns SUCCESS or FAILURE.

Examples

- Enlarges the tangram.pan 2D image by a factor 2:

```
ptrianglerescale 2 2 0 tangram.pan a.pan
```

- Shrinks the tangram.pan 2D image by a factor 2:

```
ptrianglerescale 0.5 0.5 0 tangram.pan a.pan
```

See also

Transformation, plinearrescale, pbicubicrescale, planczosrescale, pmitchellrescale, prescale

C++ prototype

```
Errc PTriangleRescale( const Img2duc &im_in, Img2duc &im_out, const float zoomy, const float zoomx );
```

Version française

Retaille d’une image par un filtre triangulaire.

Author: Régis Clouard

ptxt2col

Builds a collection from text file.

Synopsis

ptxt2col *filename* [*col_out*|-]

Description

ptxt2col creates the collection *col_out* from a value or an array of values given in the input text file. For a value, the text file has the following structure:

```
Type name value
```

For example:

```
Float pi 3.141592
```

For an array, the text file has the following structure:

```
Array:type name values
```

For example:

```
Array:Ushort iota 1 2 3 4 5 6 7
Array:Char hop 15 34 x 6 56 72 78
```

This example creates a collection with two arrays: the array `iota` that contains 7 Ushort and the array `hop` that contains 10 Char. (`34 x 6` indicates that 34 is repeated 6 times.)

Notice: strings are defined by the type `Array:Char` and follows the structure:

```
Array:Char name value
```

For example:

```
Array:Char my_string hello_word
```

Inputs

- *filename*: a regular text file.

Outputs

- *col_out*: a collection.

Result

Returns SUCCESS or FAILURE.

Examples

Creates the collection col.pan from the text file a.txt:

```
ptxt2col a.txt col.pan
```

See also

Collection, pcol2txt

C++ prototype

```
Errc PTxt2Col( const std::string &filename, Collection &col_out );
```

Version française

Construction d’une collection à partir d’un fichier texte.

Author: Alexandre Duret-Lutz

ptxt2pan

Converts text file to Pandore image file.

Synopsis

ptxt2pan *type w h d im_in* [*im_out*|-]

Description

ptxt2pan builds the new Pandore image *im_out* from the list of points from the input text file *im_in*. The result image format is defined from the parameter values. The text file must respect the followings:

```
value x y (z)
```

where z is omitted in case of 2D images.

Parameters

- *type* defines the output Pandore image type:
  - 0: Img2duc
  - 1: Img2dsl
  - 2: Img2dsf
  - 3: Img3duc
  - 4: Img3dsl
  - 5: Img3dsf
- *w*, *h* and *d* defines respectively the width, the height and the depth of the result Pandore image.

Inputs

- *im_in*: a regular text file.

Outputs

- *im_out*: Pandore image file.

Result

Returns SUCCESS or FAILURE.

Examples

Builds an 2D byte image of size 256x256 with the pixels given with the specified file image.txt:

```
ptxt2pan 0 256 256 0 image.txt image.pan
```

See also

Conversion, ppan2txt

C++ prototype

```
Errc PTxt2pan( const Img3duc &im_out, char *nom, Long d, Long h,
Long w );
```

Version française

Conversion d’une liste de points dans un fichier texte en une image.

Author: François Angot

puniformitymerging

Performs priority region merging based on uniformity criterion.

Synopsis

puniformitymerging *number threshold* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

puniformitymerging merges connected regions of the input image *rg_in* if the difference between the uniformity of the region is lower than the specified *threshold*.

Two regions are connected if there exists a link between the related nodes in the input graph *gr_in*.

The principle of the algorithm is as follows:

- For each region of the input region map *rg_in*,
- If the difference between the criterion value of the two connected regions ≤ *threshold* then merge them into one region.

The algorithm uses the priority merging that consists in merging regions with the lower difference.

The output region map *reg_out* defines the new regions and the output graph *graph_out* defines the new relationship between regions.

The uniformity criterion is calculated from:

```
uniformity(R)= 1 - ( variance(R) / (mean(R)^2) )
```

Parameters

- *number* specifies the number of allowed merging. If *number* = -1 then all possible merging are done.
- *threshold* specifies the maximum difference allowed between two regions to decide to merge them. Values are from [0..1], where 0 corresponds to non uniform regions, and 1 corresponds to highly uniform regions. A typical value is 0.95.

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.
- *gr_out*: a graph.

Result

Returns the number of merging.

Examples

Merges regions yielded by a quadtree splitting process:

```
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
```

See also

Segmentation

C++ prototype

```
Errc PUniformityMerging( const Reg2d &rg_in, const Graph2d &gr_in,
Img2duc &im_in, Reg2d &rg_out, Graph2d &gr_out, long number, float
threshold );
```

Version française

Fusion prioritaire de régions selon le critère d’uniformité.

Author: Laurent Quesnel

puniformityquadtree

Performs quadtree (or octree) segmentation based on uniformity degree.

Synopsis

puniformityquadtree *threshold* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

puniformityquadtree segments the input image *im_in* into homogeneous regions. Homogeneous regions are regions that have an inner uniformity degree ≤ *threshold*. Output regions are rectangular.

The principle of the algorithm is as follows:

- At the begin consider the image as the first block.
- If the block violates the uniformity predicate (i.e. inner uniformity ≤ *threshold*) then split the block into four equally sized sub-blocks and then apply the algorithm recursively on each sub-blocks.

Therefore, the result is composed of rectangular regions.

The uniformity degree is calculated from:

```
uniformity(R)= 1 - ( variance(R) / (mean(R)^2) )
```

For 3D image, the output region map is composed of octree regions.

Parameters

- *threshold* is the maximum uniformity value to decide if a region is homogeneous or not. Values are from the interval [0..1] where:
  - 1 corresponds to highly uniform regions.
  - 0 corresponds to non uniform regions.

Inputs

- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of regions.

Examples

Builds the quadtree of tangram.pan:

```
puniformityquadtree 0.9 tangram.pan a.pan
```

See also

Segmentation

C++ prototype

```
Errc PUniformityQuadtree( const Img2duc &im_in, Reg2d &rg_out, float threshold );
```

Version française

Segmentation d’une image par quadtree (ou octree) selon l’uniformité.

Author: Laurent Quesnel

pvalueclassnumber

Counts the number of different used values of image, region map or graph.

Synopsis

pvalueclassnumber [*-m mask*] [*im_in*|-]

Description

pvalueclassnumber counts the number of different used values in the input image *im_in*.

For image, **pvalueclassnumber** counts the number of used gray levels including value 0.

For region map, it returns the number of labels. Label 0 is not taken into account.

For graph, it returns the number of different node values including value 0.

Inputs

- *im_in*: an image, a region map or a graph.

Result

Returns the number of different pixel values in the input image *im_in*. This value can be get using operator **pstatus**.

Examples

Returns the number of grayscale used in tangram.pan:

```
pvalueclassnumber tangram.pan
pstatus
```

See also

Image Features Extraction

C++ prototype

```
Double PValueClassNumber( const Img2duc &im_in );
```

Version française

Comptage du nombre de valeurs différentes dans une image, une carte de régions ou un graphe.

Author: Régis Clouard

pvaluenumber

Returns the amount of non null pixels of image, region map or graph.

Synopsis

pvaluenumber [*-m mask*] [*im_in*|-]

Description

pvaluenumber counts the number of non null pixels in the input image *im_in*. If the input object is a graph, **pvaluenumber** counts the number of non null nodes.

Inputs

- *im_in*: a grayscale, a region map or a graph.

Result

Returns the number of non null pixels in the input image *im_in*. This value can be get using operator **pstatus**.

Examples

Returns the number of non null pixels in tangram.pan (Unix version):

```
pvaluenumber tangram.pan
val=‘pstatus‘
echo "Total = $val"
```

Returns the number of non null pixels in tangram.pan (MsDOS version):

```
pvaluenumber tangram.pan
call pstatus
call pset val
echo Total = %val%
```

See also

Image Features Extraction

C++ prototype

```
Double PValueNumber( const Img2duc &im_in );
```

Version française

Comptage du nombre de pixels non nuls dans une image (un graphe ou une carte de régions).

Author: Régis Clouard

pvaluerank

Gets the nth value for image, region map or graph.

Synopsis

pvaluerank *index* [*-m mask*] [*im_in*|*-*]

Description

pvaluerank returns the *index*th value (beginning with 1) in the input grayscale image or graph *im_in*.

This operator is useful to get the threshold value of an image yielded by a thresholding operator.

Parameters

- *index* is an integer from [1..MAX+1], where MAX depends on the type image type (e.g., 255 for a Uchar image). If the value is greater than the higher value than the higher value is used.

Inputs

- *im_in*: a grayscale image, a region map or a graph.

Result

Returns the *index*th value or FAILURE if the *index*th value does not exit. This value can be get using operator **pstatus**.

Examples

Returns the value of the first threshold yielded by a Chanda thresholding of the tangram.pan image (Unix version)

```
pchanda 10 tangram.pan a.pan
pvaluerank 1 a.pan
threshold=‘pstatus‘
```

Returns the value of the first threshold yielded by a Chanda thresholding of the tangram.pan image (MsDos version)

```
pchanda 10 tangram.pan a.pan
pvaluerank 1 a.pan
call pstatus
call pset threshold
```

See also

Image Feature Extraction

C++ prototype

```
Errc PvalueRank( const Img3duc &im_in, int index );
```

Version française

Détermination de la ieme valeur d’une image.

Author: Régis Clouard

pvarianceaggregation

Performs pixel aggregation based on variance criterion.

Synopsis

pvarianceaggregation *connexity threshold* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

pvarianceaggregation builds a new region map from aggregation of connected pixels to regions of the input region map *rg_in*. A pixel p is aggregated to a region R if:

- p is connected to the region R according to the specified *connexity*;
- |variance(R)-variance(R+im_in[p])| <= threshold

The variance degree is calculated from:

```
variance(R)= SUM((im_in[i] - mean(R))^2, i in R) / N
```

where im_in[i] are pixels of the region R, and
N is the number of pixels of the region R.

The variance of the region is not updated with the new pixel to avoid moving away too much from the initial situation. One prefer iterative executions of the operator to update the inner mean. For example, operator can be iterated until pstatus returns 0.

The output region map *rg_out* has the same number of labels than the input region map.

Parameters

- *connexity* specifies the neighbor relation between pixels (4 or 8 for 2D; 6 or 26 for 3D).
- *threshold* specifies the maximum variance value to decide to aggregate a pixel to the region. Values are from the gray scale of the input image.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of aggregation or FAILURE.

Examples

Aggregates pixels to tangram pieces:

```
pbinarization 96 1e30 tangram.pan a.pan
plabeling 8 a.pan b.pan
pvarianceaggregation 8 1 b.pan tangram.pan out.pan
```

See also

Segmentation

C++ prototype

```
Errc PVarianceAggregation( const Reg2d &rg_in, const Img2duc &im_in,
Reg2d &rg_out, int connexity, Uchar threshold );
```

Version française

Croissance des régions d’une carte selon la variance intérieure.

Author: Régis Clouard

pvariancebinarization

Performs binarization on image using interclass variance criterion.

Synopsis

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

Description

pvariancebinarization classifies pixels of the input image *im_in* into 2 clusters. The threshold value is determined as the gray level value that maximizes the interclass variance (according to Otsu's algorithm)

The principle of the algorithm is as follows:
Let h be the histogram of the gray levels, pi the probability of the gray level i and N the total number of pixels:

```
pi=h[i]/N
```

To classify the pixel into 2 clusters C0 and C1 with the threshold t:
Let p(c) and p(c1) the probabilities for a pixel x to belong respectively to the class C0 and C1:

```
p(C0) = SUM(i=0;s){ pi }
p(C1) = SUM(i=s+1;N){ pi }
```

The means for each class C0 and C1 are:

```
M0 = SUM(i=0;s){ pi/p(C0) }
M1 = SUM(i=s+1;N){ pi/p(C1) }
```

The total mean for the input image is:

```
Mt = SUM(i=0;N){ i.pi }
```

The optimal threshold maximize the interclass variance V(t):

```
V(t) = SUM(i=1;2) { P(Ci).(Mi-Mt).(Mi-Mt)
```

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).

Outputs

- *im_out*: a grayscale image of bytes (Img2duc, Img3duc).

Result

Returns the threshold value..

Examples

Segments the tangram pieces:

```
pvariancebinarization tangram.pan a.pan
```

See also

Thresholding

C++ prototype

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

Version française

Binarisation de l’image par analyse de la variance interclasse.

Reference

N. Otsu, "A threshold selection method from grey scale histogram", *IEEE Trans. on Syst. Man and Cyber*., vol 1, pp 62-66, 1979.

Author: Régis Clouard

pvariancefiltering

Performs variance filtering on image.

Synopsis

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

Description

pvariancefiltering applies a variance filter to the input image *im_in*. Each pixel of the input image is replaced by the variance of its neighbors (8 connexity in 2D or 26 connexity in 3D) if the input pixel value is between the two specified thresholds. Otherwise the output pixel is set to 0.

Parameters

- *low* and *high* specify the low and the high threshold values for the acceptable input values.

Inputs

- *im_in*: a grayscale image.

Outputs

- *im_out*: a grayscale image of reals (Img2dsf or Img3dsf).

Result

Returns SUCCESS or FAILURE.

Examples

Filters the tangram.pan:

```
pvariancefiltering 0 255 tangram.pan a.pan
```

See also

Filtering

C++ prototype

```
Errc PVarianceFiltering( const Img2duc &im_in, Img2duc &im_out,
Uchar low, Uchar high );
```

Version française

Filtrage d’une image par variance.

Author: François Angot

pvariancemerging

Performs priority region merging based on variance criterion.

Synopsis

pvariancemerging *number threshold* [*-m mask*] [*rg_in*|-] [*gr_in*|-] [*im_in*|-] [*rg_out*|-] [*gr_out*|-]

Description

pvariancemerging merges connected regions of the input image *rg_in* if the difference between the variance of the region is lower than the specified *threshold*.

Two regions are connected if there exits a link between the related nodes in the input graph *gr_in*.

The principle of the algorithm is as follows:

- For each region of the input region map *rg_in*:
  - if the difference between the criterion value of the two connected regions ≤ *threshold* then merge them into one region.

The algorithm uses the priority merging that consists in merging regions with the lower difference.

The output region map *reg_out* defines the new regions and the output graph *gr_out* defines the new relationship between regions.

The variance is calculated as follows:

```
variance(R)= SUM((im_in[i] - mean(R))^2, i in R) / N
```

where im_in[i] are pixels of the region R, and N is the number of pixels of the region R.

Parameters

- *number* specifies the number of allowed merging. If *number* = -1 then all possible merging are done.
- *threshold* specifies the maximum difference allowed between two regions to decide to merge them. Values are from the gray scale of the input image.

Inputs

- *rg_in*: a region map.
- *gr_in*: a graph.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.
- *gr_out*: a graph.

Result

Returns the number of merging.

Examples

Merges regions yielded by a quadtree splitting process:

```
puniformityquadtree 0.9 tangram.pan a.pan
prg2gr a.pan b.pan
pvariancemerging -1 45 a.pan b.pan tangram.pan c.pan d.pan
```

See also

Segmentation

C++ prototype

```
Errc PVarianceMerging( const Reg2d &rg_in, Graph2d &gr_in, Img2duc
&im_in, Reg2d &rg_out, Graph2d &gr_out, long number, float threshold
);
```

Version française

Fusion prioritaire de régions selon le critère de la variance.

Author: Laurent Quesnel

pvariancequadtree

Performs quadtree (or octree) segmentation based on variance uniformity.

Synopsis

pvariancequadtree *threshold* [*-m mask*] [*im_in*|-] [*rg_out*|-]

Description

pvariancequadtree segments the input image *im_in* into homogeneous regions. Homogeneous regions are regions that have an inner variance degree ≤ *threshold*.

The principle of the algorithm is as follows:

- At the begin consider the image as the first block.
- If the block violates the uniformity predicate (i.e. inner variance ≤ *threshold*) then split the block into four equally sized sub-blocks and then apply the algorithm recursively on each sub-blocks.

Therefore, the result is composed of rectangular regions.

The variance degree is calculated from:

```
variance(R)= SUM((im_in[i] - mean(R))^2, i in R) / N
```

where im_in[i] are pixels of the region R, and
N is the number of pixels of the region R.

For 3D image, the output region map is composed of octree regions.

Parameters

- *threshold* is the maximum variance value to decide if a region is homogeneous or not. Values are from the gray scale of the input image *im_in* (eg. 0-255 for Uchar image).

Inputs

- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of regions.

Examples

Builds the quadtree of tangram.pan:

```
pvariancequadtree 10 tangram.pan a.pan
```

See also

Segmentation

C++ prototype

```
Errc PVarianceQuadtree( const Img2duc &im_in, Reg2d &rg_out, float threshold );
```

Version française

Segmentation d’une image par quadtree (ou octree) selon la variance.

Author: Laurent Quesnel

pvarianceselection

Selects regions from variance factor.

Synopsis

pvarianceselection *relation threshold* [*-m mask*] [*rg_in*|-] [*im_in*|-] [*rg_out*|-]

Description

pvarianceselection selects regions from their variance factor. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

The region variance is calculated as follows:

```
var = ((n*sigma2)-(sigma*sigma)) / (n*n)
```

where `sigma` is the sum of gray levels of the region,
where `sigma2` is the sum of the square of the gray levels of the region,
where `n` is the number of pixels of the region.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is a real positive value.

Inputs

- *rg_in*: a region map.
- *im_in*: a grayscale image.

Outputs

- *rg_out*: a region map.

Result

Returns the number of selected regions.

Examples

Selects regions with a variance ≤ 20:

```
pvarianceselection -2 20 rin.pan rout.pan
```

See also

Region

C++ prototype

```
Errc PVarianceSelection( const Reg2d &rg_in, Img2duc &im_in, Reg2d
&rg_out, int relation, float threshold );
```

Version française

Sélection de régions sur leur valeur de variance.

Author: Régis Clouard

pvariancevalue

Measures global variance of grayscale image or graph.

Synopsis

pvariancevalue [*im_in*|-] [*col_out*|-]

Description

pvariancevalue measures the global variance of the input grayscale image or graph *im_in*.

The variance is calculated as follows:

```
variance = MOMENT_2 - (MEAN * MEAN);
```

For image, the variance is measured from the pixel values.

For graph, the variance is measured from the node values.

The variance values for each band are stored in the collection *col_out*.

Notice : This operator is not maskable.

Inputs

- *im_in*: a grayscale image or a graph.

Outputs

- *col_out*: a collection of float values.

Result

Returns the global variance value (for the first band only). This value can be get using operator **pstatus**.

Examples

Measures the global variance of the tangram.pan (Unix version):

```
pvariancevalue tangram.pan col.pan
val=‘pstatus‘
echo "Variance = $val"
```

Measures the global variance of the tangram.pan (MsDos version):

```
pvariancevalue tangram.pan col.pan
call pstatus
call pset val
echo Variance = %val%
```

See also

Image Features Extraction

C++ Prototype

```
Float PVarianceValue( const Img2duc &im_in, Collection & col_out );
```

Version française

Calcul de la valeur variance des pixels d’une image (d’un graphe ou d’une carte de régions).

Author: Régis Clouard

pversion

Prints the current Pandore version number.

Synopsis

pversion

Description

pversion prints the current Pandore version number as a string, for example:

```
PANDORE 6.0.0 (2006-04-13)
```

Result

No result value.

Examples

```
pversion
```

See also

Information

Version française

Affichage du numéro de version de la distribution Pandore.

Author: Régis Clouard

pvff2pan

Converts VFF image file to Pandore image file.

Synopsis

pvff2pan *im_in* [*im_out*|-]

Description

pvff2an converts a VFF (SunVision) image file to a 3D Pandore image file of bytes. The output image is always a 3D image of Char (Img3duc).

Inputs

- *im_in*: a VFF image file.

Outputs

- *im_out*: a 3D Pandore image file.

Result

Returns SUCCESS or FAILURE.

Examples

Converts the vff image image.vff to a Pandore image:

```
pvff2pan image.vff image.pan
```

See also

Conversion, ppan2vff

C++ prototype

```
Errc PVff2Pan( const FILE *fp, Img3duc &im_out );
```

Version française

Conversion d’une image VFF en format Pandore 3D.

Author: François Angot

pvinet

Computes the discrepancy measure between two region maps based on the number of mis-segmented pixels.

Synopsis

pvinet [*-m mask*] [*rg1_in*|-] [*rg2_in*|-]

Description

pvinet computes a discrepancy measure that accounts for the disparity between a segmented image (the region map *rg1_in*) and a reference image (the region map *rg1_in*) as defined by L. Vinet[*].

The method consists in determining each pair of regions that have a maximum overlapping ratio between the two region maps and to characterize the discrepancy by the number of pixels that do not participate to the overlapping.

The result measure is in [0..1]. The smaller is the discrepancy measure, the better is the segmentation.

Caution: Regions with label=0 are not considered for computing.

Inputs

- *rg1_in*: a region map (segmented image).
- *rg2_in*: a region map (reference image).

Result

Returns a positive real value in [0,1]. 0 means that all the regions are identical, and 1 that all the regions are totally different.
(Use `pstatus` to get this value).

Examples

Discrepancy measure between two region maps, one translated from the other:

```
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
```

See also

Evaluation

C++ prototype

```
Errc PVinet( const Reg2d &rg1_in, const Reg2d &rg2_in );
```

Version française

Calcul de la mesure de dissimilarité entre deux segmentations basée sur le nombre de pixels mal segmentés.

Reference

[*] JP. Cocquerez, S. Philipp, "*Analyse d'images: filtrage et segmentation*", Masson, 1995.

Author: Régis Clouard

pvisu

Displays Pandore object.

Synopsis

pvisu [*qt_options*] [*-m mask*] [*im_in*|-]

Description

pvisu is a graphical image display for Pandore objects. Available Pandore objects are:

- images;
- regions map;
- graphs.

The Qt version of **pvisu** also accepts Qt options at the beginning of the arguments list:

```
pvisu -style motif tangram.pan
```

All tools are accessible through menu or by shortcut key.

To visualize the contents of a collection, use pcontentsdisplays.

To draw lines onto the input image, use pdraw operator.

Inputs

- *im_in*: an image, a region map or a graph.

Result

Returns the process ID (PID) of the related process or FAILURE.

Examples

Visualizes the image tangram.pan:

```
pvisu tangram.pan
```

Visualizes the contents of the center part of the tangram.pan image (size 50x50):

```
pshapedesign 256 256 0 1 50 50 a.pan
pim2rg a.pan m.pan
pvisu -m m.pan tangram.pan
```

See also

Visualization, pdraw, pcontentsdisplay

Version française

Affichage d’un fichier de type Pandore.

Author: Régis Clouard

pvolumeselection

Selects regions from volume factor.

Synopsis

pvolumeselection *relation threshold* [*-m mask*] [*rg_in*|-] [*rg_out*|-]

Description

pvolumeselection selects regions from their volume. The parameter *relation* specifies the relation order to the *threshold* value that is used to select or not a region.

The volume is calculated from the number of voxels included in the region and on the boundary.

Parameters

- *relation* is an integer from [-3,3] which specifies the relation order to the *threshold* value:
  - *relation* = 3: regions with the maximum value.
  - *relation* = 2: regions >= *threshold*.
  - *relation* = 1: regions > *threshold*.
  - *relation* = 0: regions = *threshold*.
  - *relation* = -1: regions < *threshold*.
  - *relation* = -2: regions <= *threshold*.
  - *relation* = -3: regions with the minimum value.
- *threshold* is an integer defined in voxel unit.

Inputs

- *rg_in*: a 3D region map.

Outputs

- *rg_out*: a 3D region map.

Result

Returns the number of selected regions.

See also

Region

Examples

Selects regions with a volume = 50 voxels.

```
pvolumeselection 0 50 reg1.pan reg2.pan
```

C++ prototype

```
Errc PVolumeSelection( const Reg3d &rg_in, Reg2d &rg_out, int
relation, Ulong threshold );
```

Version française

Sélection de régions sur leur valeur de volume.

Author: Régis Clouard

pvoronoi

Calculates the Voronoi diagram.

Synopsis

pvoronoi [*-m mask*] [*rg_in*|-] [*rg_out*|-][*im_out*|-]

Description

pvoronoi builds the Voronoi diagram from the seed map *rg_in*. A seed is a pixel that has a unique label value in the region map.

Voronoi diagrams represent the region of influence around each of a given set of seeds. Every point in the region around a seed is closer to that seed than to any of the other seeds.

The algorithm uses the Chanfrein distance which is an approximation of the euclidean distance.

rg_out is the Voronoi region map where region keeps the same label value than the related seed in the input region map *rg_in*.

im_out is the distance image that results from the Voronoi diagram. Each pixel is set with the distance to the closer seed.

The Delaunay triangulation is the geometric dual of the Voronoi diagram.

Inputs

- *rg_in*: a region map.

Outputs

- *rg_out*: a region map.
- *im_out*: a signed long image.

Result

Returns SUCCESS or FAILURE.

Examples

Calculates the Voronoi diagram from the centers of mass of tangram pieces:

```
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
```

See also

Segmentation, pdelaunay.

C++ prototype

```
Errc PVoronoi( Reg &im_in, Reg2d &rg_out1, Img2dsl &im_out );
```

Version française

Calcul de la partition de Voronoï.

Author: Régis Clouard

pwatershed

Performs the watershed on image.

Synopsis

pwatershed [*-m mask*] [*rg_in*|-] [*im_pot*|-] [*rg_out*|-]

Description

pwatershed segments images into watershed regions and their boundaries. Considering the input image *im_pot* as a surface, each seeds of the input region *rg_in* can be thought of as the point to which water falling on the surrounding region drains. The boundaries of the watersheds lie on the tops of the ridges. This operator labels each watershed region with a unique index, and sets the boundaries to zero.

If the potential image *im_pot* is an image of gray levels. Each pixel value corresponds to the potential value of the point. For example, an image of distance to the objects borders or directly the image of the gray levels are acceptable potential images.
If the potential image *im_pot* is a color image, then the value of potential is the Euclidean distance between the color of the point and the average color of the area.

The principle of the algorithm is to label all the pixels that touch a seed area while beginning by those which have the lowest potential value. For that, one manages a priority file.

Note: To obtain an acceptable result, it is necessary that the seeds are minima of the potential image. For that, it can be necessary to reverse the image of potentials (see pinverse).

Inputs

- *rg_in*: a region map.
- *im_pot*: a grayscale or a color image.

Outputs

- *rg_out*: a region map.

Result

Returns SUCCESS or FAILURE.

Examples

Builds the skeleton by influence zones (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
```

See also

Morphology

C++ prototype

```
Errc PWatershed( const Reg2d &rg_in, const Img2duc &im_pot, Reg2d
&rg_out );
```

Version française

Ligne de Partage des Eaux.

Author: Abderrahim Elmoataz, Olivier Lezoray

pweszka

Performs multi-thresholding on image using Weszka algorithm.

Synopsis

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

Description

pweszka classifies the input image pixels into a small number of clusters according to their value. Every pixel p of the input image is assigned to a cluster identified by the related threshold value:

```
if threshold[k-1]<im_out[p]<=threshold[k].
then im_out[p]=threshold[k]
```

The last threshold is equal to 255.

The number of clusters and the value of the thresholds are determined from the Weszka algorithm. It is based on the measure of the busyness ratio. For each gray level i the busyness ratio is:

```
busyness(i) = SUM(SUM(Tkl)) + SUM(SUM(Tpq))
```

with k in [0..i], l in [i+1..N-1], p in [i+1..N-1] and q in [0..i].

The co-occurrence matrix Tkl contains the number of times the gray level k is a neighbor of the gray level l considering the neighborhood Nxy={ (x,y+1), (x,y-1), (x-1,y), (x+1,y) }.

Then the thresholds are located as regional minima of the busyness function. The minima are searched in the space of *length* gray levels around the gray level i.

Notice: This operator can only work on grayscale image of bytes.

Parameters

- *length* defined the length of the search space of the regional minima. It is defined in gray level unit. The greater is the length, the less there are thresholds. A typical value is 10.

Inputs

- *im_in*: a grayscale image of bytes (Img2duc, Img3duc).

Outputs

- *im_out*: a grayscale image of bytes (Img2duc, Img3duc).

Result

Returns the number of thresholds.

Examples

Segments tangram.pan and displays the number of thresholds:

```
pweszka 10 tangram.pan out.pan
pstatus
```

See also

Thresholding

C++ prototype

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

Version française

Multiseuillage de l’image par analyse de la matrice de co-occurence selon Weszka.

Reference

J.S. Weszka, "Survey of threshold selection techniques", *Computer Graphics and Image Processing*, Vol.7, pp. 259-265, 1978.

Author: Régis Clouard

pxor

Performs binary xor between images or graphs and symmetrical difference between region maps.

Synopsis

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

Description

pxor performs a bitwise "xor" between values of the two inputs *im_in1* and *im_in2*.

If inputs are integer images, the "xor" uses the '^' C operator and is applied on each pixel:

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

For real image, the "xor" operator is:

```
if pixel(im1[p]) == pixel(im[2]))
then pixel(imd[p]=0)
else pixel(imd[p]) = pixel(im1[p])+pixel(im2[p])
```

For color or multispectral image, the "xor" operator is computed separately on each band.

For graph, the "xor" operator is implemented by if then else and it is applied on each node values.

For region map, the "xor" operator is the symmetrical difference between regions:

Union(*im_in1*,*im_in2*) - Intersection(*im_in1*,*im_in2*).

The two inputs must be of the same type.

Inputs

- *im_in1*: an image, a graph or a region map.
- *im_in2*: an image, a graph or a region map.

Outputs

- *im_out*: an object of the same type as the two inputs.

Result

Returns SUCCESS or FAILURE.

For region map, returns the new higher label value.

Examples

```
pxor a.pan b.pan c.pan
```

See also

logic

C++ prototype

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

Version française

Ou exclusif binaire entre images ou graphes et différence symétrique entre cartes de régions.

Author: Régis Clouard

pxyz2lab

Converts XYZ color image to Lab color image.

Synopsis

pxyz2lab *primaries* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pxyz2lab converts color image from the color space XYZ to the color space Lab.

LAB defines 3 components:

- L est la luminancy
- a is red/blue chrominancy
- b is yellow/blue chrominancy

The conversion uses the matrix:

```
   L=116*((Y/Yn)^(1/3)) if Y/Yn>0.008856
   L=903.3*Y/Yn if Y/Yn<=0.008856
   a=500*(f(X/Xn)-f(Y/Yn))
   b=200*(f(Y/Yn)-f(Z/Zn))

where
   if Y/Yn>0.008856 f(t)=t^(1/3)
   else f(t)=7.787*t+16/116
```

Parameters

- *primaries* is an integer from [0..6] which defines the type of 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

Inputs

- *im_in*: a XYZ color image.

Outputs

- *im_out*: a Lab color image.

Result

Returns SUCCESS or FAILURE.

Examples

```
pxyz2lab a.pan b.pan
```

See also

Color

C++ prototype

```
Errc PXYZ2LAB( const Imc2dsf &im_in, Imc2dsf &im_out );
```

Version française

Changement d’espace couleur de XYZ vers Lab.

Author: Olivier Lezoray

pxyz2luv

Converts XYZ color image to L*u*v color image.

Synopsis

pxyz2luv *primaries* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pxyz2luv converts color image from the color space XYZ to the color space L*u*v (L: Luminancy, u,v : chrominancy).

Luv is very used in calculation of small colors or color differences, especially with additive colors.

```
L* = 116*((Y/Yn)^(1/3)) whether Y/Yn>0.008856
L* = 903.3*Y/Yn whether Y/Yn<=0.008856
u* = 13*(L*)*(u'-u'n)
v* = 13*(L*)*(v'-v'n)
```

where u'=4*X/(X+15*Y*+3*Z) and v'=9*Y/(X+15*Y+3*Z)
and u'n and v'n have the same definitions for u' and v' but applied to the white point reference given par the parameter *primaries*. So, you have:
u'n=4*Xn/(Xn+15*Yn*+3*Zn) and v'n=9*Yn/(Xn+15*Yn+3*Zn).

The output image is of float type.

Parameters

- *primaries* is an integer from [0..6] which defines the type of 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

Inputs

- *im_in*: a XYZ color image.

Outputs

- *im_out*: a L*u*v color image.

Result

Returns SUCCESS or FAILURE.

Examples

```
pxyz2luv 4 a.pan b.pan
```

See also

Color

C++ prototype

```
Errc PXYZ2LUV( const Imc2dsf &im_in, Imc2dsf &im_out, int primaries
);
```

Version française

Changement d’espace couleur de XYZ vers Luv.

Author: Olivier Lezoray

pxyz2rgb

Converts XYZ color image to RGB color image.

Synopsis

pxyz2rgb *primaries* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pxyz2rgb converts color image from the color space XYZ to the color space RGB.

The conversion uses the conversion matrix. For example, for the case *primaries* = 4 (illuminant C primaries NTSC):

```
|  1.910  -0.532  -0.288 |
| -0.985   1.999  -0.028 |
|  0.058  -0.118   0.898 | * 255
```

Parameters

- *primaries* is an integer from [0..6] which defines the type of 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.

Inputs

- *im_in*: a XYZ color image.

Outputs

- *im_out*: a RGB color image.

Result

Returns SUCCESS or FAILURE.

Examples

Converts parrot.pan from rgb to xyz and conversely:

```
prgb2xyz 4 parrot.pan a.pan
pxyz2rgb 4 a.pan b.pan
```

See also

Color, prgb2xyz

C++ prototype

```
Errc PXYZ2RGB( const Imc2duc &im_in, Imc2dsf &im_out, int primaries
);
```

Version française

Changement d’espace couleur de XYZ vers RGB.

Author: Régis Clouard

pyuv2pan

Converts a YUV image sequence file to a Pandore image file.

Synopsis

pyuv2pan `width height first_frame last_frame to_rgb` *`im_in`* [*`im_out`*|`-`]

Description

pyuv2pan converts a YUV image sequence file format (8 bits, encoding 4:2:0) to Pandore image file.

A YUV file stocks a sequence of color images with the color space YUV. Such image file format is uncompressed and without header. The output file *im_out* is a 3D color image (Imc3duc) or a 2D color image if the sequence is only one image (Imc2duc).

Parameters

- *width* specifies the width of the images in the sequence.
- *height* is the height of the images in the sequence.
- *first_frame* specifies the number of the first frame.
- *last_frame* specifies the number of the last frame or -1 to use the last frame.
- *to_rgb* specifies the if the color space is RGB if to_rgb=1 or YUV format if to_rgb=0.

Inputs

- *im_in*: a YUV image sequence (8 bits, encoding 4:2:0).

Outputs

- *im_out*: a Pandore image (Imc2duc or Imc3duc).

Result

Returns SUCCESS or FAILURE.

Examples

Converts all the image sequence image.yuv to a 3D Pandore image with color space RGB:

```
pyuv2pan 256 256 0 -1 1 image.yuv image.pan
```

See also

Conversion

C++ prototype

```
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);
```

Version française

Conversion d’un fichier de séquence d’images au format YUV (8 bits, codage 4:2:0) en un fichier Pandore.

Important notice

Author: David Tschumperlé

pyuv2rgb

Converts YUV color image to RGB color image.

Synopsis

pyuv2rgb [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pyuv2rgb converts color image from the color space YUV (television standard) to the color space RGB (Red, Gren, Blue).

YUV is adopted by the PAL television standard.

The color conversion is a linear transformation:

```
|  1   0          0.402    |
|  1  -0.344136  -0.714136 |
|  1   0          1.772    |
```

Inputs

- *im_in*: a YUV color image.

Outputs

- *im_out*: a RGB color image.

Result

Returns SUCCESS or FAILURE.

Examples

```
prgb2yuv parrot.pan a.pan
pyuv2rgb a.pan b.pan
```

See also

prgb2yuv, Color

C++ prototype

```
Errc PYUV2RGB( const Imc2duc &im_in, Imc2dsf &im_out );
```

Version française

Changement d’espace couleur de RVB vers YUV.

Author: Meftah Boudjelal

pzeboudj

Computes the goodness measure based on inter and intra-region contrast.

Synopsis

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

Description

pzeboudj computes a goodness measure for quantitative evaluation of gray level image segmentation results as defined by R. Zeboudj [*].

The measure is based on the inter and intra-region contrast.
Contrast of the pixel s with its neighbor t in the image I is measured as follows:

```
   c(s,t) = | I(s)- I(t) | / L-1
```

with

```
       L is max(ims)-min(ims).
```

The inner contrast of region Ri is:

```
Ii=1/Ai * sum_Ri  [ max{c(s,t), t in W(s) inter Ri} ]
```

The outer contrast of region Ri is:

```
Ei=1/li * sum Fi [ max{c(s,t), t in W(s), t not in Ri} ]
```

where Fi is the boundary of Ri and li the length of Fi.

The contrast of region Ri is:

```
C(Ri) = { 1 - Ii/Ei if 0 < Ii < Ei;
        { Ei        if Ii=0;
        { 0         otherwise;
```

Finally, the global contrast is:

```
Contrast = 1/A * sum [Ai.c(Ri)]
```

The result is a value in [0..1]. The higher the value of the Zeboudj measure is, the better the segmentation result should be.

Caution: Regions with label=0 are not considered for computing.

Inputs

- *rg_in*: a region map.
- *im_in*: a gray scale image.

Result

Returns a positive real value.
(Use `pstatus` to get this value).

Examples

Computes the zeboudj measure for a simple binarization segmentation process:

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

See also

Evaluation

C++ prototype

```
Errc PZeboudj( const Reg2d &rg_in, const Img2duc &im_in );
```

Version française

Calcul du critère de qualité basé sur le contraste inter et intra-régions.

Reference

* JP. Cocquerez, S. Philipp, "*Analyse d'images: filtrage et segmentation*", Masson, 1995.

Author: Régis Clouard

pzerocross

Locates zero crossing.

Synopsis

pzerocross *connexity value* [*-m mask*] [*im_in*|-] [*im_out*|-]

Description

pzerocross detects the place where the pixel is a transition between two neighbor pixels. A transition is detected when:

- A pixel value is greater than *value* and at least one of its neighbors is lower than *value*,
- A pixel value is lower than *value* and at least one of its neighbors is greater than *value*.

The output image *im_out* is a unsigned char grayscale image (Img2duc or Img3duc).

This operator is mainly used to detect zero crossing of the laplacian. For example, the value is equal to 127 for unsigned char images and 0 for signed long image.

Parameters

- *connexity* defines the neighbor relationship: 4, 8 in 2D or 6, 26 in 3D. This parameter is ignored for graph.
- *value* belongs to gray levels of the input image *im_in*.

Inputs

- *im_in*: a grayscale image.

Outputs

- *im_out*: a binary image.

Result

Returns SUCCESS or FAILURE.

Examples

Performs an edge detection using the DOG algorithm (Difference Of Gaussian):

```
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
```

See also

Edge detection

C++ prototype

```
Errc PZeroCross( const Img2duc &im_in, Img2duc &im_out, int
connexity, Uchar value );
```

Version française

Localisation des changements de signe des valeurs de pixels.

Author: Régis Clouard