Image Toolbox Manual

Chapter 2

Operator Descriptions

Chapter 2 - Operator Descriptions

A. Introduction

This chapter contains the man1 pages for all programs in the Image toolbox, organized alphabetically by object name. Any man1 page can be accessed by typing the following command.

```
% kman <program_name>
```

This information can also be accessed by selecting the *Help* button on the program's graphical user interface, which is displayed when running the program from within cantata, or when running it from the command line using the *-gui* option.

The keywords (-k) option in kman allows the user to provide a keyword string for searching. When invoked with the *-k* option, kman will perform the search and provide a list of the names and short descriptions of programs found by the search.

```
% kman -k "keyword string"
```

If you wish to display only the command line usage information for a program, enter the program name, followed by -U.

```
% <program_name> -U
```

B. Image Processing Operators

B.1. ibpf — *2-Dimensional Band-Pass Filter Design*

Object Information

Category:	*Image Proc*	Subcategory:	*Frequency Filter*
Operator:	*Band-Pass*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

This routine generates a spatial frequency image of real values that represents the response of a Band-Pass Filter.

The generator function brands are Ideal, Butterworth, Exponential, Chebyshev I, and Chebyshev II. All output is radially symmetric about the origin of the image, and the origin can be either the center of the image or the top left corner of the image. Cutoff frequencies are normalized between 0 and 0.5 where 0 is DC, and 0.5 is the maximum circle/ellipse that will cover the image.

The filters are all radially symmetric about an origin that can be located at pixel (0,0) or at (image_width/2, image_height/2). If image_height and image_width are not the same value, the distance formula will normalize your distanced from the origin like by this formula.

d = sqrt[(x/image_width)^2 + (y/image_height)^2]

This implies that a non-square image will produce an ellipse frequency image as a result.

All filter types (Low-Pass, High-Pass, Band-Pass, and Band-Stop) use the Low-Pass transfer function and a frequency transform formula on the distance returned by the formula above (References listed below).

Examples

None

See Also

ifilt_design, lifilt_design, kimage_proc

Restrictions

None

References

The algorithms and formulas for this routine was taken from several sources. From Samuel D. Stearns and Don Hush, "Digital Signal Analysis", Prentice Hall 1990; chapter 12, "Analog and Digital Filter Design", provided the chebyshev equations (page 266) and a table of frequency transforms (table 12.1 on page 281). Rafael Gonzalez and Paul Wintz, "Digital Image Processing", 2nd Ed., Addison-Wesley 1987 provided the Butterworth equations in Chapter 4, page 170. The exponential transform function was taken from the low pass filter routines in Khoros 1.0.

Command Line Arguments

```
Usage for ibpf: 2-Dimensional Band-Pass Filter Design
% ibpf
  -o          (outfile) Output frequency image filename

  Required Mutually Exclusive Group - Specify exactly one of the 5 options:
    [-ideal] (flag) Ideal transfer response
    [-but]   (flag) Butterworth transfer response
    [-exp]   (flag) Exponential transfer response
    [-ch2]   (flag) Chebyshev II transfer response
    [-ch1]   (flag) Chebyshev I transfer response

  [-pos]      (cycle)   Low frequencies located at image center or corners
          0  "Corners of Image",
          1  "Center of Image"
          [default: 1 "Center of Image"]
  [-type]     (cycle)   Data Type of frequency image to output
          10  "Float",
          11  "Double"
```

```
        [default: 11 "Double"]
[-order]   (integer) Order of filter to generate
           (order >= 0, default = 1)
[-lcutoff] (double)  Normalized lower band cutoff frequency
           (0 <= lcutoff <= 0.5, default = 0.2)
[-ucutoff] (double)  Normalized upper band cutoff frequency
           (0 <= ucutoff <= 0.5, default = 0.3)
[-hsize]   (integer) The height size of the filter
           (hsize > 0, default = 256)
[-wsize]   (integer) The width size of the filter
           (wsize > 0, default = 256)
```

B.2. ibsf — *2-Dimensional Band-Stop Filter Design*

Object Information

Category:	*Image Proc*	Subcategory:	*Frequency Filter*
Operator:	*Band-Stop*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

This routine generates a spatial frequency image of real values that represents the response of a Band-Stop Filter.

The generator function brands are Ideal, Butterworth, Exponential, Chebyshev I, and Chebyshev II. All output is radially symmetric about the origin of the image, and the origin can be either the center of the image or the top left corner of the image. Cutoff frequencies are normalized between 0 and 0.5 where 0 is DC, and 0.5 is the maximum circle/ellipse that will cover the image.

The filters are all radially symmetric about an origin that can be located at pixel (0,0) or at (image_width/2, image_height/2). If image_height and image_width are not the same value, the distance formula will normalize your distanced from the origin like by this formula.

d = sqrt[(x/image_width)^2 + (y/image_height)^2]

This implies that a non-square image will produce an ellipse frequency image as a result.

All filter types (Low-Pass, High-Pass, Band-Pass, and Band-Stop) use the Low-Pass transfer function and a frequency transform formula on the distance returned by the formula above (References listed below).

Examples

None

See Also

ifilt_design, lifilt_design, kimage_proc

Restrictions

None

References

The algorithms and formulas for this routine was taken from several sources. From Samuel D. Stearns and Don Hush, "Digital Signal Analysis", Prentice Hall 1990; chapter 12, "Analog and Digital Filter Design", provided the chebyshev equations (page 266) and a table of frequency transforms (table 12.1 on page 281). Rafael Gonzalez and Paul Wintz, "Digital Image Processing", 2nd Ed., Addison-Wesley 1987 provided the Butterworth equations in Chapter 4, page 170. The exponential transform function was taken from the low pass filter routines in Khoros 1.0.

Command Line Arguments

```
Usage for ibsf: 2-Dimensional Band-Stop Filter Design
% ibsf
  -o         (outfile) Output frequency filter filename

  Required Mutually Exclusive Group - Specify exactly one of the 5 options:
    [-ideal] (flag) Ideal transfer response
    [-but]   (flag) Butterworth transfer response
    [-exp]   (flag) Exponential transfer response
    [-ch1]   (flag) Chebyshev I transfer response
    [-ch2]   (flag) Chebyshev II transfer response

  [-pos]      (cycle)   Low frequencies located at image center or corners
          0  "Corners of Image",
          1  "Center of Image"
          [default: 1 "Center of Image"]
  [-type]     (cycle)   Type of frequency image to output
          10  "Float",
          11  "Double"
          [default: 11 "Double"]
  [-order]    (integer) Order of filter to generate
              (order >= 0, default = 1)
  [-lcutoff]  (double)  Normalized lower band cutoff frequency
              (0 <= lcutoff <= 0.5, default = 0.2)
  [-ucutoff]  (double)  Normalized upper band cutoff frequency
              (0 <= ucutoff <= 0.5, default = 0.3)
  [-hsize]    (integer) The height size of the output filter
              (hsize > 0, default = 256)
  [-wsize]    (integer) The width size of the output filter
              (wsize > 0, default = 256)
```

B.3. ifilt_design — *General 2-Dimensional Image Filter Design*

Object Information

Category:	*Image Proc*	Subcategory:	*Frequency Filter*
Operator:	*Design Filter*	Object Type:	*kroutine*
In Cantata:	*No*		

Description

This routine generates a spatial frequency image of real values that represents the response of the filter type and brand that was specified. The types it supports are Low Pass, High Pass, Band Pass, and Band Stop filters.

The generator function brands are Ideal, Butterworth, Exponential, Chebyshev I, and Chebyshev II. All output is radially symmetric about the origin of the image, and the origin can be either the center of the image or the top left corner of the image. Cutoff frequencies are normalized between 0 and 0.5 where 0 is DC, and 0.5 is the maximum circle/ellipse that will cover the image.

The filters are all radially symmetric about an origin that can be located at pixel (0,0) or at (image_width/2, image_height/2). If image_height and image_width are not the same value, the distance formula will normalize your distanced from the origin like by this formula.

d = sqrt[(x/image_width)^2 + (y/image_height)^2]

This implies that a non-square image will produce an ellipse frequency image as a result.

All filter types (Low-Pass, High-Pass, Band-Pass, and Band-Stop) use the Low-Pass transfer function and a frequency transform formula on the distance returned by the formula above (References listed below).

Examples

ifilt_design -o output.filt -lp -but -order 4 -ucutoff 0.1

This command will generate a 256 x 256 image of doubles values with a range from 0 to 1 that corresponds to the frequency image that results from a normalized cutoff of 0.1 and the Butterworth transfer function. The DC value is centered at 128, 128 in the image.

See Also

lifilt_design, kimage_proc

Restrictions

None

References

The algorithms and formulas for this routine was taken from several sources. From Samuel D. Stearns and Don Hush, "Digital Signal Analysis", Prentice Hall 1990; chapter 12, "Analog and Digital Filter Design", provided the chebyshev equations (page 266) and a table of frequency transforms (table 12.1 on page 281). Rafael Gonzalez and Paul Wintz, "Digital Image Processing", 2nd Ed., Addison-Wesley 1987 provided the Butterworth equations in Chapter 4, page 170. The exponential transform function was taken from the low pass filter routines in Khoros 1.0.

Command Line Arguments

```
Usage for ifilt_design: General 2-Dimensional Image Filter Design
% ifilt_design
  -o         (outfile) Output frequency filter image

  Required Mutually Exclusive Group - Specify exactly one of the 4 options:
    [-lp] (flag) Low Pass
    [-hp] (flag) High Pass
    [-bp] (flag) Band Pass
    [-bs] (flag) Band Stop

  Required Mutually Exclusive Group - Specify exactly one of the 5 options:
    [-ideal] (flag) Ideal filter
    [-but]   (flag) butterworth
    [-exp]   (flag) exponential
    [-ch1]   (flag) chebyshev1
    [-ch2]   (flag) chebyshev2

  [-pos]      (cycle)   Low frequencies located at image center or corners
          0  "Corners of Image",
          1  "Center of Image"
          [default: 1 "Center of Image"]
  [-type]     (cycle)   Type of frequency image to output
          10  "Float",
          11  "Double"
          [default: 11 "Double"]
  [-order]    (integer) Filter order
              (order >= 0, default = 1)
  [-lcutoff] (double)  Normalized lower cutoff frequency
              (0 <= lcutoff <= 0.5, default = 0.25)
  [-ucutoff] (double)  Normalized upper cutoff frequency
              (0 <= ucutoff <= 0.5, default = 0.25)
  [-hsize]    (integer) Image height size
              (hsize > 0, default = 256)
  [-wsize]    (integer) Image width size
              (wsize > 0, default = 256)
```

B.4. igamut — *Compress Color Multiband Image to Single Band with Map*

Object Information

Category:	*Image Proc*	Subcategory:	*Data Compression*
Operator:	*Compress Colors*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

igamut takes a 3 band multi-spectral image, where each band would normally represent one component of a 24-bit color image (RGB, for example), and generates a pseudo color image with a specified number of colors that can be displayed. The object is to make the quantized image look like the original 24-bit image even though the number of colors is greatly restricted.

The quantization is performed by isolating clusters of "neighboring" color vectors in a three dimensional histogram, with each axis being one of the color components. The clusters are obtained using a modified version of Heckbert's median cut. The true colors are then matched to the closest cluster, and the input RGB triplet is then re-mapped to an n-color pseudo color image.

To keep the histogram from becoming exceedingly large (max of around 2^24 bytes), one may need to quantize the grey levels of the input bands to less than 8 bits. 6 bits (64 levels) gives results that are reasonable in a short amount of time. The number of bits that are kept is called the color precision, which can be specified at execution time. The general tradeoff is that smaller precision is faster and takes less memory, but it looks worse too. High precision takes longer and great gobs of memory, but looks decent, provided that a reasonable number (say 128 or more) colors is specified. The execution time is very dependent on the image statistics. In general, a small number of colors is faster than a large number of colors. In either case, if the image has good spatial color coherence, execution time is greatly reduced.

The allocation fraction (-r) controls how large areas of nearly the same color are handled. An allocation fraction of 0.0 will cause the large areas to be broken into as many colors as possible with the largest areas of a particular color range being broken first. An allocation fraction of 1.0 will attempt to preserve the detail in the image by preserving the color range of all parts of the image at the expense of smooth coloring of the larger areas. An allocation fraction of around 0.2 to 0.5 gives very good results on most images.

As a special note, be sure that the input image has a map that has one-to-one correspondence to the pixel values (i.e. the pixel value is the grey level). If this is not the case, use kmapdata(1) to convert to a linear map.

If the input image contains less than the number of colors requested then the output image will contain only the number of colors present in the input image. The color map will contain the number of entries requested (meaning all colors in the image) with any extra entries zero padded.

No dithering is done by *igamut.* Dithering in color space can often cause the image to look very noisy due to the color sensitivity of the eye. The result images from *igamut* may display some banding effects (Mach bands), but these are often more tolerable visually than the noise that would result from a color dithering process. (Should someone be suddenly inspired, there is enough information stored in the internal data structures to permit detection of Mach bands and automatic suppression of these artifacts; a few hundred more lines of C code should suffice to accomplish this.) Adding a small amount of noise to the image before using *igamut* can reduce Mach band effects greatly.

If the KINFO environment variable is set to VERBOSE, then an additional diagnostic message, the number of colors actually found in the image, will be printed. The color count is subject to the color precision specified by the (-p) option.

Multiple plane images are processed by quantizing each plane independently, generating a corresponding plane of colors in the map. Thus an input object with (w,h,d,t,e)=(512,480,10,10,4) will result in an output object with a value segment with dimensions (512,480,10,10,1) and a mpa segment with dimensions (4,10,10,10,1).

Examples

% igamut -i lizard.rgb.viff -o lizard.240.viff -n 240 -p 8
quantizes the colors present in the 24-bit RGB input file down to 240 colors, using all 8 bits of each band of the input image.

See Also

kmapdata(1)

References

The reference on which this routine is based is: P. Heckbert, "Color Image Quantization for Frame Buffer Display", Computer Graphics, Vol. 16, No. 3, July 1982, p297.

Command Line Arguments

```
Usage for igamut: Compress Color Multiband Image to Single Band with Map
% igamut
  -i         (infile)  Multiband input image (elements dimension > 1)
  -o         (outfile) output file

  [-n]       (integer) Number of colors to compress down to
             (n > 0, default = 240)
  [-p]       (integer) Number of bits of precision to use during quantization
             (0 <= p <= 32, default = 8)
  [-r]       (float)   Fraction of color splits based on subspace 2-norm
             (0 <= r <= 1, default = 0.5)
```

B.5. igauss_func — *Generate Gaussian Function Image*

Object Information

Category:	*Input/Output*	Subcategory:	*Generate Data*
Operator:	*2D Gaussian*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

igauss_func generates an image containing one or more 2-dimensional Gaussian functions. The image has dimensions of wsize*hsize and can have one of the following types,**bit, byte, unsigned byte, short, unsigned short, integer, unsigned integer, long, unsigned long, float** and **double**. The parameters for the Gaussian function can be either specified directly if only one Gaussian is desired. If multiple Gaussians are desired, then the parameters are specified as a multiband image. Six parameters need to be specified for each Gaussian function to be generated. These are: x and y coordinate of the peak of the function, variance along x and y direction, correlation coefficient and the amplitude of the Gaussian function.

The correlation coefficient can range between -1 and 1. It represents the degree of correlation between the Gaussian functions along the 2 dimensions. A value of 1 implies perfect correlation whereas a value of 0 implies that the two variables are independent of each other.

For the multiple Gaussian case, the data vector of the parameter image must contain 6 elements denoting the parameters for the 2D Gaussian function, in the order indicated above. If the normalization flag is set, the amplitude value for the Gaussian function is ignored.

If the parameter image contains the map segment, then the data is mapped through that segment before being read for supplying parameters. An error occurs if the parameter image contains a mask segment or if the parameter image does not have 6 bands of data.

Command Line Arguments

```
Usage for igauss_func: Generate Gaussian Function Image
% igauss_func
  -o          (outfile) Output file containing Gaussian data

  Required Mutually Exclusive Group - Specify exactly one of the 2 options:

    [Optional Mutually Inclusive Group - Specify all or none of the 6 options:]
      [-wpeak]    (integer) Width coordinate for peak of 2D Gaussian function
                  (unbounded, default = 256)
      [-hpeak]    (integer) Height coordinate for peak of 2D Gaussian function
                  (unbounded, default = 256)
      [-wvar]     (double)  Variance of the 2D Gaussian function along width
                  (wvar > 0.0, default = 1)
      [-hvar]     (double)  Variance of the 2D Gaussian function along height
                  (hvar > 0.0, default = 1)
      [-coeff]    (double)  Correlation coefficient for 2D Gaussian function
                  (-1 <= coeff <= 1, default = 0)
      [-peakval]  (double)  Peak value of the 2D Gaussian function
                  (unbounded, default = 1)

    [Optional Mutually Inclusive Group - Specify all or none of the 2 options:]
      [-i]     (infile)  Input file of parameters for multiple Gaussian functions
               (default = {none})
      [-norm]  (boolean) Normalize amplitude of the Gaussian function
               (default = true)

  [-wsize]    (integer) Width of output image (number of columns)
              (wsize > 0, default = 512)
  [-hsize]    (integer) Height of output image (number of rows)
              (hsize > 0, default = 512)
  [-type]     (list)    Data type of 2D Gaussian function data
          1  "bit",
          2  "byte",
          3  "unsigned byte",
          4  "short",
          5  "unsigned short",
          6  "integer",
          7  "unsigned integer",
          8  "long",
          9  "unsigned long",
          10  "float",
          11  "double"
          [default: 10 "float"]
```

B.6. igeowarp — *Perform Direct Bilinear Geometric Warping*

Object Information

Category:	*Image Proc*	Subcategory:	*Geometric Operators*
Operator:	*GeoWarp*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

igeowarp is used to apply a direct bilinear warping to an image. In this mode, the bilinear mapping equations give the mapping of output data coordinates to input data coordinates. A given output data coordinate is operated on by the mapping functions to give the input data coordinate. The value of the input data coordinate is interpolated using bilinear interpolation, and the resulting value is assigned to the output data coordinate.

All warping is assumed to occur only in the WxH plane. If warping in a different plane is desired, it is necessary to use the kaxis program to reorient the data such that the desired plane is the WxH plane before applying igeowarp. This limitation is due to computational complexity and performance issues, particularly when dealing with large data sets.

The mapping functions used by igeowarp are:

f(w,h) = output image

g(w',h') = input image coordinate

w' = a00 + a01*w + a10*h +a11*w*h
h' = b00 + b01*w + b10*h +b11*w*h

Note that in general, w' and h' are not integers for a given integer pair w and h. The data value at g(w',h') is obtained by bilinear interpolation of the four nearest neighbors to coordinate (w',h').

The mapping is called direct because it directly gives the location in the input data that corresponds to a given location in the output data. The opposite case gives the location in the output data that corresponds to a given input data location; this requires much trickier and computationally expensive interpolation methods to obtain good results.

The input parameters a00, a01, a10, a11, b00, b01, b10, b11 are exactly as specified in the mapping functions above. These parameters can also be supplied in the form of object value data where the data is assumed to be organized with the parameters a00, a01, a10, and a11 present in that storage order for the W coefficients and likewise for the H coefficients.

If the input object has a mask, a new mask will be computed for the output object indicating which data points contain reliable data.

If the input object has a map, the data is pulled through the map prior to rotation, and the output object will have no map.

The -planes flag is used to change the way the data is accessed for processing. If the data set is small enough that individual planes of data (including the E data) parallel to the WxH plane will fit in memory, then use of the -planes flag will permit processing to happen in a plane-by-plane basis; this can occur with efficiency and speed. Otherwise, processing will occur in prisms down the D axis, which is much slower but will work on data sets of any size. If you have lots of memory, you may be able to get away with using -planes even for quite large data sets, say around 2Kx2K or more points per plane.

See Also

irotate(1)

Restrictions

For this initial release, the E dimension data is *not* handled correctly. Data in WxHxD *is* handled correctly. This deficiency will be corrected in the next version of *igeowarp.*

Command Line Arguments

```
Usage for igeowarp: Perform Direct Bilinear Geometric Warping
% igeowarp
  -i         (infile)  Input data object
  -o         (outfile) Output data object

  Required Mutually Exclusive Group - Specify exactly one of the 2 options:
    [-wcoeffs] (infile) W warp function coeffs object
               (default = {none})

    [Optional Mutually Inclusive Group - Specify all or none of the 4 options:]
      [-a00] (double) A00 (W warp function, constant term)
             (-100 <= a00 <= 100, default = 0)
      [-a01] (double) A01 (W warp function, w term)
             (-100 <= a01 <= 100, default = 0)
      [-a10] (double) A10 (W warp function, h term)
             (-100 <= a10 <= 100, default = 0)
      [-a11] (double) A11 (W warp function, wh term)
             (-100 <= a11 <= 100, default = 0)

  Required Mutually Exclusive Group - Specify exactly one of the 2 options:
    [-hcoeffs] (infile) H warp function coeffs object
               (default = {none})

    [Optional Mutually Inclusive Group - Specify all or none of the 4 options:]
      [-b00] (double) B00 (H warp function, constant term)
             (-100 <= b00 <= 100, default = 0)
      [-b01] (double) B01 (H warp function, w term)
             (-100 <= b01 <= 100, default = 0)
      [-b10] (double) B10 (H warp function, h term)
             (-100 <= b10 <= 100, default = 0)
      [-b11] (double) B11 (H warp function, wh term)
             (-100 <= b11 <= 100, default = 0)

  [-planes]  (flag)    Process data by full planes instead of small prisms
```

B.7. igradient — *Gradient Operators (Roberts, Sobel, Prewitt, Isotrophic)*

Object Information

Category:	*Image Proc*	Subcategory:	*Spatial Filters*
Operator:	*Gradient Operator*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

igradient takes an image and performs the differentiation (high pass spatial filtering or edge enhancement) operation by using standard gradient operators. The standard gradient operators supported by the *igradient* routine are: Robert's, Sobel, Prewitt and Isotropic). The Robert's operators use a 2x2 kernel, whereas the other three use a 3x3 kernel. Each operator uses two templates to enhance edges in the horizontal and vertical directions only. For edge enhancements in other directions other operators should be used.

There are two approximations to the true gradient. The user can select between calculating the magnitude of the gradient by using squares and square roots (Euclidean distance evaluation), or by using absolute differences. The latter is computationally faster.

Mask data is ignored.

Examples

% igradient -i source_image.viff -o dest_image.viff
performs edge enhancement using the default values, i.e., uses the Sobel templates and uses absolute differences to approximate the gradient.

% igradient -i source_image.viff -o dest_image.viff -isotropic -abs 1
performs edge enhancement using the Isotropic templates and determines the gradient by evaluating Euclidean distances.

Command Line Arguments

```
Usage for igradient: Gradient Operators (Roberts, Sobel, Prewitt, Isotrophic)
% igradient
  -i         (infile)  input image filename
  -o         (outfile) output image filename

  Required Mutually Exclusive Group - Specify exactly one of the 4 options:
    [-roberts]   (flag) Perform Roberts gradient operation
    [-sobel]     (flag) Perform Sobel gradient operation
    [-prewitt]   (flag) Perform Prewitt gradient operation
    [-isotropic] (flag) Perform Isotropic gradient operation

  [-abs]     (boolean) Calculate gradient using absolute value or square root
             (default = false)
```

B.8. ihpf — *2-Dimensional High-Pass Filter Design*

Object Information

Category:	*Image Proc*	Subcategory:	*Frequency Filter*
Operator:	*High-Pass*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

This routine generates a spatial frequency image of real values that represents the response of a High-Pass Filter.

The generator function brands are Ideal, Butterworth, Exponential, Chebyshev I, and Chebyshev II. All output is radially symmetric about the origin of the image, and the origin can be either the center of the image or the top left corner of the image. Cutoff frequencies are normalized between 0 and 0.5 where 0 is DC, and 0.5 is the maximum circle/ellipse that will cover the image.

The filters are all radially symmetric about an origin that can be located at pixel (0,0) or at (image_width/2, image_height/2). If image_height and image_width are not the same value, the distance formula will normalize your distanced from the origin like by this formula.

d = sqrt[(x/image_width)^2 + (y/image_height)^2]

This implies that a non-square image will produce an ellipse frequency image as a result.

All filter types (Low-Pass, High-Pass, Band-Pass, and Band-Stop) use the Low-Pass transfer function and a frequency transform formula on the distance returned by the formula above (References listed below).

Examples

None

See Also

ifilt_design, lifilt_design, kimage_proc

Restrictions

None

References

The algorithms and formulas for this routine was taken from several sources. From Samuel D. Stearns and Don Hush, "Digital Signal Analysis", Prentice Hall 1990; chapter 12, "Analog and Digital Filter Design", provided the chebyshev equations (page 266) and a table of frequency transforms (table 12.1 on page 281). Rafael Gonzalez and Paul Wintz, "Digital Image Processing", 2nd Ed., Addison-Wesley 1987 provided the Butterworth equations in Chapter 4, page 170. The exponential transform function was taken from the low pass filter routines in Khoros 1.0.

Command Line Arguments

```
Usage for ihpf: 2-Dimensional High-Pass Filter Design
% ihpf
  -o         (outfile) Output frequency filter filename

  Required Mutually Exclusive Group - Specify exactly one of the 5 options:
    [-ideal] (flag) Ideal transfer response
    [-but]   (flag) Butterworth transfer response
    [-exp]   (flag) Exponential transfer response
    [-ch1]   (flag) Chebyshev I transfer response
    [-ch2]   (flag) Chebyshev II transfer response

  [-pos]      (cycle)   Low frequencies located at image center or corners
          0  "Corners of Image",
          1  "Center of Image"
          [default: 1 "Center of Image"]
  [-type]     (cycle)   Type of frequency image to output
          10  "Float",
          11  "Double"
          [default: 11 "Double"]
  [-order]    (integer) Order of filter to generate
              (order >= 0, default = 1)
  [-lcutoff] (double)  Normalized lower cutoff frequency
              (0 <= lcutoff <= 0.5, default = 0.25)
  [-hsize]    (integer) The height size of the output filter
              (hsize > 0, default = 256)
  [-wsize]    (integer) The width size of the output filter
              (wsize > 0, default = 256)
```

B.9. ilpf — *2-Dimensional Low-Pass Filter Design*

Object Information

Category:	*Image Proc*	Subcategory:	*Frequency Filter*
Operator:	*Low-Pass*	Object Type:	*pane*
In Cantata:	*Yes*		

Description

This routine generates a spatial frequency image of real values that represents the response of a Low-Pass Filter.

The generator function brands are Ideal, Butterworth, Exponential, Chebyshev I, and Chebyshev II. All output is radially symmetric about the origin of the image, and the origin can be either the center of the image or the top left corner of the image. Cutoff frequencies are normalized between 0 and 0.5 where 0 is DC, and 0.5 is the maximum circle/ellipse that will cover the image.

The filters are all radially symmetric about an origin that can be located at pixel (0,0) or at (image_width/2, image_height/2). If image_height and image_width are not the same value, the distance formula will normalize your distanced from the origin like by this formula.

d = sqrt[(x/image_width)^2 + (y/image_height)^2]

This implies that a non-square image will produce an ellipse frequency image as a result.

All filter types (Low-Pass, High-Pass, Band-Pass, and Band-Stop) use the Low-Pass transfer function and a frequency transform formula on the distance returned by the formula above (References listed below).

Examples

None

See Also

ifilt_design, lifilt_design, kimage_proc

Restrictions

None

References

The algorithms and formulas for this routine was taken from several sources. From Samuel D. Stearns and Don Hush, "Digital Signal Analysis", Prentice Hall 1990; chapter 12, "Analog and Digital Filter Design", provided the chebyshev equations (page 266) and a table of frequency transforms (table 12.1 on page 281). Rafael Gonzalez and Paul Wintz, "Digital Image Processing", 2nd Ed., Addison-Wesley 1987 provided the Butterworth equations in Chapter 4, page 170. The exponential transform function was taken from the low pass filter routines in Khoros 1.0.

Command Line Arguments

```
Usage for ilpf: 2-Dimensional Low-Pass Filter Design
% ilpf
  -o         (outfile) Output frequency filter filename

  Required Mutually Exclusive Group - Specify exactly one of the 5 options:
    [-ideal] (flag) Ideal transfer response
    [-but]   (flag) Butterworth transfer response
    [-exp]   (flag) Exponential transfer response
    [-ch1]   (flag) Chebyshev I transfer response
    [-ch2]   (flag) Chebyshev II transfer response

  [-pos]     (cycle)   Low frequencies located at image center or corners
          0  "Corners of Image",
          1  "Center of Image"
          [default: 1 "Center of Image"]
  [-type]    (cycle)   Type of frequency image to output
          10  "Float",
          11  "Double"
```

```
        [default: 11 "Double"]
[-order]   (integer) Order of filter to generate
           (order >= 0, default = 1)
[-ucutoff] (double)  Normalized upper cutoff frequency
           (0 <= ucutoff <= 0.5, default = 0.25)
[-hsize]   (integer) The height size of the output filter
           (hsize > 0, default = 256)
[-wsize]   (integer) The width size of the output filter
           (wsize > 0, default = 256)
```

B.10. imedian — *Median Filter for Image Data*

Object Information

Category:	*Image Proc*	Subcategory:	*Nonlinear Filters*
Operator:	*Median*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

imedian performs median filtering on image data. The method is as follows: for each WSIZExHSIZE window in the image, replace the pixel at (WSIZE/2)x(HSIZE/2) (integer arithmetic) with the median of the values in the window. The median is computed by sorting the values in the window with quicksort and extracting the middle value as the median. This is not the fastest approach for BYTE data where a histogram updating method could be used. It is, however, extensible to higher data types including floating point data.

If a map is present, the data is mapped prior to operation. Mask data is ignored.

This implementation processes the image data by full planes.

Boundaries are handled by zero padding.

Restrictions

Complex data types are not supported.

The ability to process by partial planes will be added at a future time, permitting more efficient operation on images large in the W and H dimensions.

At some point, the faster histogram updating technique should be stolen from K1.5 lvhmed(3) and put in place for BYTE processing.

Command Line Arguments

```
Usage for imedian: Median Filter for Image Data
% imedian
  -i         (infile)  Input image
  -o         (outfile) Output image
```

```
[-wsize]    (integer) Width of window
            (wsize > 0, default = 3)
[-hsize]    (integer) Heigth of window
            (hsize > 0, default = 3)
[-reps]     (integer) number of repetitions of filter to perform
            (reps >= 0, default = 1)
```

B.11. ipostscr — *Convert Image to Postscript*

Object Information

Category:	*Input/Output*	Subcategory:	*Hardcopy Output*
Operator:	*Postscript*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

ipostscr formats an image for output on a Postscript printer. The width and height are specified in inches. The X and Y offset specifies the location of the lower left corner of the image from the lower left corner of the page.

ipostscr will generate either a binary, grayscale, truecolor image from the input object. If the data set contains a map, then the first three columns (assuming that there are at least three) will be interpreted a the red, green, and blue components and will be used to construct a truecolor postscript image by mapping the value data through the map. If a map exists, but it does not have three or more columns, then the existing columns will be padded out to three columns with 0's.

If the data set does not contain a map, then if its data type is KBIT, it will generate a bitmapped black and white postscript. Otherwise, the data set will be cast to unsigned byte and a grayscale image will be generated.

This program can be used to generate either Postscript or Encapsulated Postscript images in either Landscape or Portrait mode. The -eps flag is used to specify that encapsulated postscript should be output.

If you only specify the width, or the height, the other dimension will be calculated, so that you will get a postscript image which reflects the true dimensions of the image.

Command Line Arguments

```
Usage for ipostscr: Convert Image to Postscript
% ipostscr
  -i          (infile)  Object to be output in postscript form
  -f          (outfile) output postscript file

  [-width]    (double)  Desired width in inches of the printed image
              (width > 0.0, default = 4)
  [-height]   (double)  Desired height of printed image in inches
              (height > 0.0, default = 4)
```

```
[-xoffset]  (double)  Width offset in inches from bottom left corner of printed page
            (xoffset > 0.0, default = 1)
[-yoffset]  (double)  Height offset in inches from bottom left corner of printed page
            (yoffset > 0.0, default = 1)
[-showpage] (flag)    Issue showpage at end of output

[Optional Mutually Exclusive Group - Specify one or none of the 2 options:]
  [-portrait]  (flag) Specify that portrait style output is desired.
  [-landscape] (flag) specifies that landscape output is desired

[Optional Mutually Exclusive Group - Specify one or none of the 2 options:]
  [-ps]  (flag) Specify that regular Postscript output is desired.
  [-eps] (flag) Specify that Encapsulated Postscript is desired.
```

B.12. irotate — *Rotate Object by Arbitrary Angle*

Object Information

Category:	*Image Proc*	Subcategory:	*Geometric Operators*
Operator:	*Rotate*	Object Type:	*kroutine*
In Cantata:	*Yes*		

Description

irotate is used to rotate the contents of the value segment of a data object by an arbitrary angle about an axis parallel to the DEPTH axis. Interpolation of output data from the input data is bilinear.

The rotation is performed under that assumption that all data in the WxHxD space is to be physically rotated to a new position, and that E data associated with each point in WxHxD is to be rotated along with the point itself.

If rotation about a different axis is desired, it is necessary to use the kaxis program to reorient the data such that the desired axis is the DEPTH axis before applying irotate. This limitation is due to computational complexity and performance issues, particularly when dealing with large data sets.

If the input object has a mask, a new mask will be computed for the output object indicating which data points contain reliable data.

If the input object has a map, the data is pulled through the map prior to rotation, and the output object will have no map.

The rotation angle is CW for a positive angle, and should be specified in degrees.

The center of rotation is used to specify that point in the input object about which the rotation is to take place. The -wctr and -hctr flags can be used to ask irotate to automatically place the center of rotation at the center of the object.

The -resize flag will causes the output object dimensions to be increased or decreased sufficiently to just contain all of the input data even after rotation. When this flag is used, the W and H dimensions may increase or decrease by a factor of up to sqrt(2). If -resize is selected and the center of rotation is not in the center of the data, the resulting image is not centered in the output data. Note that if this were the case, then there would be no need for specifying a center of rotation at all since the input data would always appear centered in the output data.

The -planes flag is used to change the way the data is accessed for processing. If the data set is small enough that individual planes of data (including the E data) parallel to the WxH plane will fit in memory, then use of the -planes flag will permit processing to happen in a plane-by-plane basis; this can occur with efficiency and speed. Otherwise, processing will occur in prisms down the D axis, which is much slower but will work on data sets of any size. If you have lots of memory, you may be able to get away with using -planes even for quite large data sets, say around 2Kx2K or more points per plane.

See Also

igeowarp(1)

Restrictions

For this initial release, the E dimension data is *not* handled correctly. Data in WxHxD *is* handled correctly. This deficiency will be corrected in the next version of *irotate.*

Command Line Arguments

```
Usage for irotate: Rotate Object by Arbitrary Angle
% irotate
  -i         (infile)  Input data object
  -o         (outfile) Output data object

  Required Mutually Exclusive Group - Specify exactly one of the 2 options:
    [-wc]    (double) W coordinate of center of rotation
             (unbounded, default = 0)
    [-wctr]  (flag)   center of rotation in center of W coordinate span

  Required Mutually Exclusive Group - Specify exactly one of the 2 options:
    [-hc]    (double) H coordinate of center of rotation
             (unbounded, default = 0)
    [-hctr]  (flag)   center of rotation in center of H coordinate span

  [-angle]   (double)  rotation angle (degrees, positive is CW)
             (-360 <= angle <= 360, default = 0)
  [-resize]  (flag)    resize to fit rotated image
  [-planes]  (flag)    Process data by full planes instead of small prisms
```

B.13. itexture — *Texture Feature Extraction Using LAW Metrics*

Object Information

Category:	*Image Proc*	Subcategory:	*Feature Extraction*
Operator:	*Texture*	Object Type:	*kroutine*
In Cantata:	*No*		

Description

itexture will extract texture features from an image by convolving the original image with a spatial filter. The spatial filters are comprised of 5 x 5 kernels derived from center-weighted vectors defined by the LAWS texture measure.

Each texture measure calculated will be stored as width-height planes along the elements direction in the Output Object (o). The order in which the (selected) texture measures will be stored in the output object are:

L5L5
L5E5
L5S5
L5W5
L5R5
E5L5
E5E5
E5S5
E5W5
E5R5
S5L5
S5E5
S5S5
S5W5
S5R5
W5L5
W5E5
W5S5
W5W5
W5R5
R5L5
R5E5
R5S5
R5W5
R5R5

If the input object elements dimension is greater than 1 then the results obtained by applying all texture kernels to the first band are stored by planes in the output object, followed by the results obtained by applying all texture kernels to the second band, etc.

The five center-weighted vectors are:

L5 = [1 4 6 4 1]
E5 = [-1 -2 0 2 1]
S5 = [-1 0 2 0 -1]
W5 = [-1 2 0 -2 1]
R5 = [1 -4 6 -4 1]

Each 5 x 5 kernel is derived from multiplying a particular combination of two of the above vectors. This results in 25 possible 5 x 5 kernels. Note that 10 of the kernels are formed by taking the transpose.

For example, L5E5 is a 5 x 5 kernel formed by multiplying the vectors, L5 with E5. It's transpose is E5L5, and is similarly formed by multiplying the vectors, E5 with L5. One of these kernels is more sensitive to horizontal changes in texture, while the other is sensitive to vertical changes in texture.

The sum of the elements of each kernel is zero, which results in the output image having a mean of zero. Therefore the relevant texture information is contained in the image variance of the microtexture features. The LAW kernels were designed to be sensitive to structures such as edges, ripples, and spots.

Choosing which kernel to use will depend on the nature of the texture of interest, and will require trial and error to decide which produces the feature with the most discriminating power. In general, if a texture kernel of say, L5S5 is used, one may also want to use its transpose, S5L5 for certain types of quasiperiodic variations commonly found in textured images.

Once the texture features are extracted, macrostatistical estimates may be obtained using a larger window size (eg. 15 x 15 or 31 x 31). Some useful statistics may be the variance or the absolute average. Such features of the texture filtered image were referred to as texture energy measures by K.I. LAWS.

Restrictions

itexture will operate on input objects of any data storage type.

The output object will be of type KLONG, KDOUBLE, or KDCOMPLEX, determined by the data type of the input object. If the input object is of data type KBYTE, KUBYTE, KSHORT, or KUSHORT, then it is converted up to an KLONG image. If the input is of type KFLOAT or KDOUBLE, then the result will be KDOUBLE. Any complex input type is converted to type KDCOMPLEX.

References

K. I. Laws, "Rapid texture identification" Proc. SPIE, vol. 238, pp.376-380, 1980.

J. Y. Hsiao, and A. A. Sawchuk, "Supervised textured image segmentation using feature smoothing and probabilistic relaxation techniques" IEEE Trans. Pattern Anal. Machine Intell., vol. 11, No. 12, pp 1279-1292, 1989.

Command Line Arguments

```
Usage for itexture: Texture Feature Extraction Using LAW Metrics
% itexture
  -i         (infile)  input object
  -o         (outfile) output object


  [Optional Loose Group - Specify at least one of the 25 options:]
    [-L5L5] (flag) If set, apply L5L5 texture kernel
    [-E5L5] (flag) If set, apply E5L5 texture kernel
    [-S5L5] (flag) If set, apply S5L5 texture kernel
    [-W5L5] (flag) If set, apply W5L5 texture kernel
    [-R5L5] (flag) If set, apply R5L5 texture kernel
    [-L5E5] (flag) If set, apply L5E5 texture kernel
    [-E5E5] (flag) If set, apply E5E5 texture kernel
    [-S5E5] (flag) If set, apply S5E5 texture kernel
    [-W5E5] (flag) If set, apply W5E5 texture kernel
    [-R5E5] (flag) If set, apply R5E5 texture kernel
```

```
[-L5S5] (flag) If set, apply L5S5 texture kernel
[-E5S5] (flag) If set, apply E5S5 texture kernel
[-S5S5] (flag) If set, apply S5S5 texture kernel
[-W5S5] (flag) If set, apply W5S5 texture kernel
[-R5S5] (flag) If set, apply R5S5 texture kernel
[-L5W5] (flag) If set, apply L5W5 texture kernel
[-E5W5] (flag) If set, apply E5W5 texture kernel
[-S5W5] (flag) If set, apply S5W5 texture kernel
[-W5W5] (flag) If set, apply W5W5 texture kernel
[-R5W5] (flag) If set, apply R5W5 texture kernel
[-L5R5] (flag) If set, apply L5R5 texture kernel
[-E5R5] (flag) If set, apply E5R5 texture kernel
[-S5R5] (flag) If set, apply S5R5 texture kernel
[-W5R5] (flag) If set, apply W5R5 texture kernel
[-R5R5] (flag) If set, apply R5R5 texture kernel
```

Table of Contents

This page left intentionally blank