1

Functions by Page

2

3

Function Listing by Purpose

4

5

Function Listing by Chapter

6

7

8

anis

Purpose

This function estimates the Anisotropy metric for a point process based on a single sample.

Synopsis

```
A = anis(X,d)
[A,fr] = anis(X,d)
[A,fr] = anis(X,d,f)
[A,fr] = anis(X,d,f,s)
[A,fr,c] = anis(X,...)
```

Description

`A = anis(X,d)` calculates the Anisotropy metric, **`A`** of type **`double`**, from the sample image, **`X`** of type **`double`**, using periodograms of dimension **`d`** by **`d`**, where **`d`** is a scalar, with spectral rings of radial width equal to one pixel.

`[A,fr] = anis(X,d)` returns the inner radii of the spectral rings in the vector **`fr`** of type **`double`**.

`[A,fr] = anis(X,d,f)` calculates the power spectrum of **`X`** using **`d`** by **`d`** windows transformed into **`f`** by **`f`** windows where **`f`** is a scalar. The parameter **`f`** may be empty or unspecified. In either case,

9

f defaults to the value specified for **d**.

[A,fr] = anis(X,d,f,s) calculates the Anisotropy metric where **s** is the amount of overlap between windows used to calculate the power spectrum estimate. The parameter **s** may be empty or unspecified. In either case, **s** defaults to the value 0.

[A,fr,c] = anis(X,...) returns the number of windows used to calculate the power spectrum estimate in **c**.

Example

The following code creates a sample FM halftone pattern and then uses the **anis** function to estimate the corresponding anisotropy. The results are then plotted versus the inner radius of each spectral ring.

```
X=ones(128,1280)*0.125;
H=halftone(X, 'rwerror');
[A, fr]=anis(double(H), 128, [], []);
plot(fr, A);
```

References

R. A. Ulichney, *Digital Halftoning*, MIT Press, Cambridge, MA, 1987.

R. A. Ulichney, Dithering with blue noise, *Proceedings of the IEEE*, 76(1):56-79, January 1988.

10

See Also

`ddf, pc, rapsd`

11

bayer

Purpose

This function halftones a continuous-tone image to binary using Bayer s dither array to create a dispersed-dot ordered dither.

Synopsis

```
H = bayer(X)
```

Description

H = bayer(X) halftones the continuous-tone image **X** using Bayer s dither array. **X** may be a color image such that **H** is the halftone created by applying Bayer s dither array to each channel separately.

Example

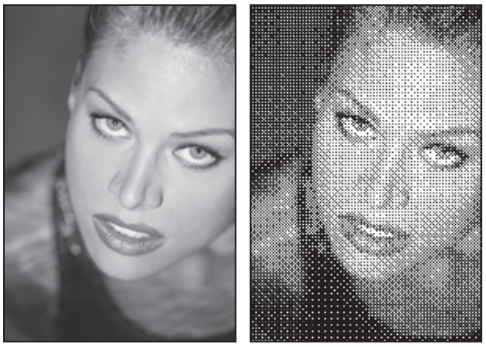
The following code converts a continuous-tone RGB image to a binary RGB image and displays the image using the imshow command.

```
load womanRGB;
H=bayer(X);
imshow(double(H));
```

References

B. E. Bayer, An optimal method for two level rendition of continuous-tone picture, *Conference Record of the IEEE International Conference on Communications*, Seattle, Washington, USA, pp. 11-15,

12



The RGB image *woman* (left) before and (right) after halftoning with **bayer**.

June 11-13, 1973.

See Also

`hlfmsk, cdod`

13

function reference

bippcca

Purpose

This function generates a binary dither pattern using the BInary-Pattern-Pair-Correlation-Construction-Algorithm.

Synopsis

```
H = bippcca(N,g,asoc)
H = bippcca(N,g,asoc,X)
H = bippcca(N,g,asoc,X,C)
H = bippcca(N,g,asoc,X,C,Var)
H = bippcca(N,g,asoc,X,C,Var,seed)
```

Description

`H = bippcca(N,g,asoc)` generates a binary dither pattern of size **`N`** (1-by-1 or 2-by-1 vector) representing gray level **`g`** with clusters of average size **`asoc`** pixels.

`H = bippcca(N,g,asoc,X)` uses **`X`** as the initial or starting dither pattern such that **`X`** is a subset of the resulting pattern **`H`**. **`X`** can be an empty set.

`H = bippcca(N,g,asoc,X,C)` uses **`C`** as a constraining dither pattern such that **`H`** is a subset of **`C`**. **`C`** can be an empty set.

14

`H = bippcca(N,g,asoc,X,C,Var)` speficifies the variance of the low pass filter used to determine the concentration matrix in the BIPPCCA algorithm. **`Var`** can be an empty set.

`H = bippcca(N,g,asoc,X,C,Var,seed)` sets the **`seed`** value for the random number generator used in the BIPPCCA algorithm.

Example

The following code creates a 64-by-128 sample green-noise dither pattern representing gray level 0.25 with an average cluster size of 9 pixels. The resulting dither pattern is then plotted using the imshow command.

```
H=bippcca([64 128], 0.25, 9);
imshow(H);
```

References

D. L. Lau, G. R. Arce, and N. C. Gallagher, Digital halftoning via green-noise masks, *Journal of the Optical Society of America A*, 16(7):1575-1586, July 1999.

See Also

`bippsma`

15

function reference

bippsma

Purpose

This function generates a blue-noise binary dither pattern using the BInary-Pattern-Power-Spectrum-Matching-Algorithm.

Synopsis

```
H = bippsma(N,g)
H = bippsma(N,g,w)
H = bippsma(N,g,w,seed)
```

Description

`H = bippsma(N,g)` generates a blue-noise binary dither pattern of size **`N`**-by-**`N`** (**`N`** scalar) representing gray level **`g`**.

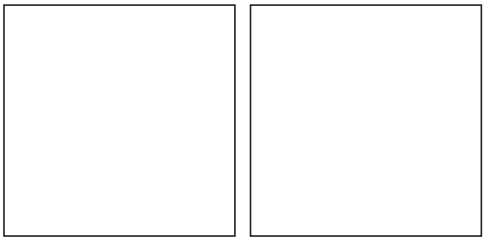
`H = bippsma(N,g,w)` uses **`w`** as the weight associated with the sigma parameter for the Gaussian Filter used in BIPPSMA.

`H = bippsma(N,g,w,seed)` uses **`seed`** to set the seed value for the random number generator used to create the initial pattern.

Example

The following code creates a 128-by-128 sample green-noise dither pattern representing gray level 0.25 with an average cluster size of 9 pixels. The resulting dither pattern is then plotted using the imshow command.

16



The initial and final dither pattern produced using **`bippsma`**.

```
H=bippsma(128, 0.25);
imshow(H);
```

References

T. Mitsa and K. J. Parker, Digital halftoning technique using a blue noise mask, *Journal of the Optical Society of America A*, 9(8):1920-1929, August 1992.

M. Yao and K. J. Parker, Modified approach to the construction of a blue noise mask, *Journal of Electronic Imaging*, 3(1):92-97, January

17

1994.

See Also

`bippsma`

18

bnm

Purpose

This function generates a Blue Noise Mask using the BIPPSMA Algorithm under the stacking constraint.

Synopsis

```
Y = bnm(N,G)
Y = bnm(N,G,M)
Y = bnm(N,G,M,s)
```

Description

`Y = bnm(N,G)` creates a blue-noise mask, using **`bippsma`**, of size **`N`**-by-**`N`** (**`N`** scalar) under the stacking constraint such that **`Y`** is composed of the gray-levels specified in **`G`**. **`G`** also specifies the order to which patterns are constructed such the the *i*th constructed pattern represents gray-level **`G`**(i). The first two values in **`G`** must be 0 and 255.

`Y = bnm(N,G,M)` generates the blue noise mask where **`M`** is a 3-dimensional array containing a stack of preconstructed dither patterns, serving as a secondary constraint imposed upon the BIPPSMA Algorithm. Any slice of **`M`** that contains all zeros is ignored for that level by BIPPSMA (no constraint). **`M`** may be empty.

`Y = bnm(N,G,M,s)` specifies the seed, **`s`**, used for the random

19

number generator for constructing initial patterns in BIPPSMA.

Example

The following code creates a 64-by-64 pixel blue-noise mask such that the 0.5 dither pattern is a checkerboard .

```
G=[0 255 [128:-1:1] [129:1:254]];
M=zeros(64,64,length(G));
 [R,S]=meshgrid(1:64);
M(:,:,find(G==0.5))=rem(R+S,2);
Y=bnm(64,G,M,1);
imshow(Y);
```

References

*See Also

`gnm, bippsma, bippcca`

20

cdod

Purpose

This function halftones a continuous-tone image using clustered-dot ordered dithering or AM halftoning.

Synopsis

```
H = cdod(X,M,N,A)
```

Description

`H = cdod(X,M,N,A)` converts the continous-tone image **`X`**, type double, to the halftone image **`H`**, type **`uint8`** (**`logical`**), using clustered-dot ordered dithering with halftone cells of size **`M`**-by-**`N`** at a screen angle of **`A`** (in degrees). If **`X`** is a monochrome image, **`cdod`** creates a pattern of clustered zeros on a background of ones; otherwise, **`cdod`** creates clusters of ones on a background of zeros.

If **`X`** is a multi-channel image, **`A`** can be a vector of length **`size(X,3)`** such that **`A`**(i) specifies the screen angle used to halftone channel i.

Example

The following code converts an CMYK continuous-tone image using AM halftoning and then plots the results using imshow command.

```
load womanCMYK;
```

21

The CMYK image *woman* (left) before and (right) after halftoning with **cdod** using 4-by-4 pixel cells (16 intensity levels) and screen angles of 15, 75, 45, and 0 degrees.

```
H=cdod(X, 4, 4, [15 75 0 45]);
imshowCMYK(H);
```


22

References

R. A. Ulichney, *Digital Halftoning*, MIT Press, Cambridge, MA, 1987.

H. McGilton and M. Campione, *PostScript by Example*, Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1992.

See Also

`halftone, errdif, verrdif`

23

cmy2cmyk

Purpose

This function converts a continuous-tone Cyan-Magenta-Yellow image to a Cyan-Magenta-Yellow-blacK image.

Synopsis

```
Y = cmy2cmyk(X)
```

Description

`Y = cmy2cmyk(X)` converts a continuous-tone CMY image, of type *double* with intensity values ranging from 0 to 1, to a continuous-tone CMYK image.

See Also

`cmy2rgb, cmyk2cmy, rgb2cmy`

24

cmy2rgb

Purpose

This function converts a continuous-tone Cyan-Magenta-Yellow image to a Red-Green-Blue image.

Synopsis

```
Y = cmy2rgb(X)
```

Description

`Y = cmy2rgb(X)` converts a continuous-tone CMY image, of type *double* with intensity values ranging from 0 to 1, to a continuous-tone RGB image.

See Also

`cmy2cmyk, rgb2cmy, cmyk2cmy`

25

cmyk2cmy

Purpose

This function converts a continuous-tone Cyan-Magenta-Yellow-blacK image to a Cyan-Magenta-Yellow image.

Synopsis

```
Y = cmyk2cmy(X)
```

Description

`Y = cmyk2cmy(X)` converts a continuous-tone CMYK image, of type *double* with intensity values ranging from 0 to 1, to a continuous-tone CMY image.

See Also

`cmyk2rgb, rgb2cmyk, cmy2cmyk`

26

cmyk2rgb

Purpose

This function converts a continuous-tone Cyan-Magenta-Yellow-blacK image to a Red-Green-Blue image.

Synopsis

```
Y = cmyk2rgb(X)
```

Description

`Y = cmyk2rgb(X)` converts a continuous-tone CMYK image, of type *double* with intensity values ranging from 0 to 1, to a continuous-tone RGB image.

See Also

`cmyk2cmy, rgb2cmyk, cmy2cmyk`

27

ddf

Purpose

This function estimates the directional distribution function for a point process based on a single sample.

Synopsis

```
T = ddf(X,r1,r2)
T = ddf(X,r1,r2,N)
[T,a] = ddf(X,...)
```

Description

`T = ddf(X,r1,r2)` calculates the directional distribution function for the point process sample **`X`**, type **`uint8`**, for the annular ring with inner radius **`r1`** and outer radius **`r2`**. **`r1`** and **`r2`** must be the same size and each can be a vector such that T is a matrix with each column corresponding to the direction distribution function for the corresponding annular ring.

`T = ddf(X,r1,r2,N)` calculates the directional distribution function for an annular ring divided into **`N`** equal length arcs such that each arc is 360/**`N`** degrees. If not specified, the default value for **`N`** is 16.

`[T,a] = ddf(X,...)` returns the center angle to each of **`N`** arcs that divide the annular rings specified by **`r1`** and **`r2`**.

28

Example

The following code creates a sample FM halftone pattern and then uses the **`ddf`** function to estimate the corresponding direction distribution function for an annular ring divided into 32 equal segments. The results are then plotted versus the center angle of each arc on a polar axis.

```
X=ones(128,128)*0.125;
H=halftone(X, 'rwerror');
[D, a]=ddf(H, 1.5, 2.5, 32);
polar(a, D);
```

References

D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic Geometry and Its Applications*, John Wiley and Sons, New York, 1987.

D. L. Lau, G. R. Arce, and N. C. Gallagher, Green-noise digital halftoning, *Proceedings of the IEEE*, 86(12):2424-2444, December 1998.

See Also

`anis, pc, rapsd`

29

dotdif

Purpose

This function converts a continuous-tone image into binary using Knuth s dot diffusion algorithm.

Synopsis

```
H = dotdif(X,M)
[H,E] = dotdif(X,M)
```

Description

`H = dotdif(X,M)` converts the continuous-tone image **`X`** into the binary halftone image **`H`** using Knuth s dot diffusion. Pixels, within a halftone cell, are processed in the sorted order of pixels in the class matrix, **`M`**. For color images, dot diffusion is applied to each channel such that the class matrix is tiled along the color axis. This implementation does not allow error to diffuse into neighboring cells.

Example

The following code halftones an RGB image using three unique 32-by-32 white-noise class matrices:

```
load womanRGB;
mask=randn(32,32,3);
H=dotdif(X, mask);
imshow(double(H));
```


30

The RGB image *woman* (left) before and (right) after halftoning with **`dotdif`** using a 32-by-32-by-3 random class matrix.

References

D. E. Knuth, Digital halftones dot dif fusion, *ACM Transactions on Graphics*, 6:245-273, October 1987.

31

M. Mese, P. P. Vaidyanathan, Improve Dot Dif fusion for Image Halftoning, *IS&T s NIP15: 1999 International Conference on Digital Printing Technologies*, Orlando, Florida USA, pp. 350-353, October 17-22, 1999.

See Also

`errdif, hlfmsk`

32

errdif

Purpose

This function converts a continuous-tone image to binary using the error diffusion algorithm with optional hysteresis term and optional threshold modulation for edge enhancement term.

Synopsis

```
H = errdif(X,W)
H = errdif(X,W,h)
H = errdif(X,W,h,d)
H = errdif(X,W,h,d,P,T)
[H,E] = errdif(X,...)
```

Description

`H = errdif(X,W)` converts the continuous-tone image **`X`**, type **`double`**, using error diffusion with the error filter specified by **`W`**.

The matrix **`W`** specifies the error and hysteresis filter weights using the form:

```
W=[a a a a a
   a a a a a
   a a k b b
   b b b b b
   b b b b b];
```

33

where **b** represents an error filter coefficient and **a** represents a hysteresis filter coefficient. The parameter **k** represents the scalar constant used to modulate the quantization threshold according to the current input pixel such the edges are enhanced in the output image **H**.

H = errdif(X,W,h) employs output-dependent feedback with a hysteresis constant equal to **h** (scalar). The hysteresis filter is specified along with the error filter in **W**.

H = errdif(X,W,h,d) uses the serpentine raster if **d=-1** (default) or a normal left-to-right raster when **d=+1**.

H = errdif(X,W,h,d,P,T) uses filter weight perturbation such that filter weights are paired according to **P** and the amount of perturbation is specified by **T**. The matrices **P** and **T** are the same size as W and are of the form of:

```
P=[ 1  2  1  3  4      T=[1 1 1 1 1
    3  5  4  5  6         1 1 1 1 1
    6  2  0  7  8         1 1 0 1 1
    9  8 10  9 10         1 1 1 1 1
   11  7 12 11 12];       1 1 1 1 1]*0.25;
```

such that weights with equal values in **P** are paired and the amount of perturbation is equal to the corresponding values in **T** times the smaller of the two weights.

34

H = errdif(X,W,h,d,P,T,s) uses **s** as the seed to the random number generator used for perturbing filter weights.

[H,E] = errdif(X,...) returns the error image **E**.

Example

The following code converts a continuous-tone image using error diffusion with the hysteresis using the original two error by two hysteresis weight kernel proposed by Levien with 50% perturbation of the weights.

```
load womanGREY;
W=[0 1 0;1 0 1;0 1 0]/2;
P=[0 1 0;1 0 2;0 2 0];
T=[0 1 0;1 0 1;0 1 0]/2;
H=errdif(X, W, 1.0, -1, P, T, 1.0);
imshow(H);
```

References

R. Levien, Output dependant feedback in error dif fusion halftoning, In *IS&T s Eighth International Congress on Advances in Non-Impact Printing Technologies*, pages 280-282, Williamsburg, Virginia, USA, October 25-30 1992.

D. L. Lau, G. R. Arce, and N. C. Gallagher, Green-noise digital halftoning, *Proceedings of the IEEE*, 86(12):2424-2444, December 1998.

35

The gray-scale image *woman* (left) before and (right) after halftoning with errdif using Levien s filter weight arrangement with perturbed filter weights.

See Also

`cdod, halftone, verrdif`

36

gnm

Purpose

This function generates a Green Noise Mask using the BIPPCCA Algorithm under the stacking constraint.

Synopsis

```
Y = gnm(N,G,asoc)
Y = gnm(N,G,asoc,M)
Y = gnm(N,G,asoc,M,s)
```

Description

Y = gnm(N,G,asoc) creates a green-noise mask, using **bippcca**, of size **N**-by-**N** (**N** scalar) under the stacking constraint such that **Y** is composed of the gray-levels specified in **G** with an average cluster size of **asoc**. **G** also specifies the order to which patterns are constructed such the the i*th* constructed pattern represents gray-level **G**(i) with clusters of average size **asoc**(i) pixels. The first two values in **G** must be 0 and 255.

Y = bnm(N,G,asoc,M) generates the green-noise mask where **M** is a 3-dimensional array containing a stack of preconstructed dither patterns, serving as a secondary constraint imposed upon BIPPCCA. Any slice of **M** that contains all zeros is ignored for that level by BIPPCCA

37

(no constraint). **M** may be empty.

Y = bnm(N,G,asoc,M,s) specifies the seed, **s**, used for the random number generator for constructing patterns in BIPPCCA.

Example

The following code creates a 64-by-64 pixel green-noise mask.

```
G=[0 255 [128:-1:1] [129:1:254]];
asoc=interp1([0 128 255],[5 20 5],G);
Y=gnm(64,G,asoc);
imshow(Y);
```

References

D. L. Lau, G. R. Arce, and N. C. Gallagher, Digital halftoning via green-noise masks, *Journal of the Optical Society of America A*, 16(7):1575-1586, July 1999.

See Also

`bnm, bippcca, bippsma`

38

hlfmsk

Purpose

This function converts a continuous-tone image to binary using a user supplied dither array of mask.

Synopsis

```
H = hlfmsk(X,M)
```

Description

H = hflmsk(X,M) converts the continuous-tone image **X**, type **double** or **uint8**, using the halftone mask **M**, of the same type as **X**. If **X** is a color image, **M** is tiled along the color axis.

Example

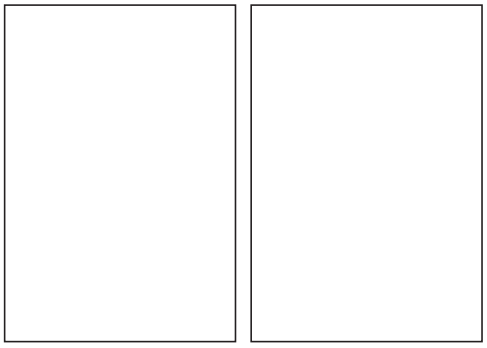
The following code converts a continuous-tone CMYK image into a color halftone using a four channel green-noise mask.

```
load womanRGB;
load gnmRGB;
H=hlfmsk(X,double(mask)/255);
imshow(H);
```

References

D. L. Lau, G. R. Arce, and N. C. Gallagher, Digital halftoning

39



The CMYK image *woman* (left) before and (right) after halftoning using a four channel green-noise mask.

via green-noise masks, *Journal of the Optical Society of America A*, 16(7):1575-1586, July 1999.

D. L. Lau, G. R. Arce, and N. C. Gallagher, Digital color halftoning

40

via generalized error-diffusion and multi-channel green-noise masks, *IEEE Transactions on Image Processing*, 9(5), May 2000.

T. Mitsa and K. J. Parker, Digital halftoning technique using a blue noise mask, *Journal of the Optical Society of America A*, 9(8):1920-1929, August 1992.

M. Yao and K. J. Parker, Modified approach to the construction of a blue noise mask, *Journal of Electronic Imaging*, 3(1):92-97, January 1994.

See Also

`bnm, bippcca, bippsma, gnm`

41

hvs

Purpose

This functions takes a continous-tone image and based on the viewing distance and the image resolution, returns an image modeling the apparent image as defined by the human visual system model.

Synopsis

```
Y = hvs(X,v,d)
```

Description

`Y = hvs(X,v,d)` returns the apparent image as defined by a model of the human visual system with a viewing distance of **`v`** (in inches) and a print resolution of **`d`** (in pixels per inch).

Example

The following code creates the apparent image seen and displays it using the imshow command.

```
X=imread('test.tif', 'tif');
Y=hvs(X, 20, 600);
imshow(Y);
```

References

F. W. Campbell, R. H. Carpenter, and J. Levinson, Visibility of

42

aperiodic patterns compared with that of sinusoidal gratings, *The Journal of Physiology*, 190:283-298, 1969.

F. W. Campbell, J. J. Kulikowski, and J. Levinson, The effect of orientation on the visual resolution of gratings, *The Journal of Physiology*, 187:427-436, 1966.

S. Daly, Subroutine for the generation of a two dimensional human visual contrast sensitivity function, *Eastman Kodak Technical Report*, (233203Y), 1987.

43

imshowCMYK

Purpose

This function displays CMY or CMYK images using Matlab s standard imshow command.

Synopsis

```
h = imshowCMYK(X);
```

Description

`h = imshowCMYK(X)` displays the CMY or CMYK image **`X`** and returns the handle to the axis in **`h`**.

See Also

`imshow`

44

intnsty

Purpose

This function calculates the intensity of points within a point process sample.

Synopsis

```
I = intnsty(X)
```

Description

`I = intnsty(X)` calculates the intensity of points (pixels set to one) in the point process sample **`X`** or type uint8 (logical).

Example

The following code calculates the intensity of minority pixels within a white-noise dither pattern:

```
X=rand(128,128)>0.25;
I=intnsty(pp(X));
```

References

D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic Geometry and Its Applications*, John Wiley and Sons, New York, 1987.

D. L. Lau, G. R. Arce, and N. C. Gallagher, Green-noise digital halftoning, *Proceedings of the IEEE*, 86(12):2424-2444, Dec. 1998.

45

See Also

`pp, rsmm, pc, anis`

46

mbippcca

Purpose

The Multichannel-Binary-Pattern-Pair-Correlation-Construction-Algorithm is used to create color dither patterns for a constant color with specific pair correlations between and within channels.

Synopsis

```
BP = vbippcca(H,g,r,R1,R2,...,Rc,F1,F2,...,Fc)
BP = vbippcca(H,g,r,R1,...,Rc,F1,...,Fc,M)
```

Description

`BP = vbippcca(H,g,r,R1,R2,...,Rc,F1,F2,...,Fc)` generates a binary dither pattern **`BP`** with intensity **`g`** using initial dither pattern **`H`** such that **`BP`** has a pair correlation as defined by the pair correlation shaping functions **`R1,R2,...,Rc`** versus **`r`** with homogeneity ensured by the low-pass filter vectors **`F1,F2,...,Fc`** versus **`r`** or by scalars **`F1,F2,...,Fc`** corresponding to the variance of a Gaussion low-pass filter.

`BP = vbippcca(H,g,r,R1,...,Rc,F1,...,Fc,M)` generates the binary dither pattern **`BP`** with the mask **`M`** such that only those pixels in **`BP`** corresponding to pixels in **`M`** set to 1 are possible candidates to become minority pixels.

47

Example

The following code creates a 128-by-128 pixel green-noise dither pattern representing CMYK color [1 1 1 1]/4.

```
G=[0 255 [128:-1:1] [129:1:254]];
M=zeros(64,64,length(G));
[R,S]=meshgrid(1:64);
M(:,:,find(G==0.5))=rem(R+S,2);
Y=bnm(64,G,M,1);
imshow(Y);
```

References

D. L. Lau, G. R. Arce, and N. C. Gallagher, Digital halftoning via green-noise masks, *Journal of the Optical Society of America A*, 16(7):1575-1586, July 1999.

D. L. Lau, G. R. Arce, and N. C. Gallagher, Digital color halftoning via generalized error-diffusion and multi-channel green-noise masks, *IEEE Transactions on Image Processing*, 9(5), May 2000.

See Also

`bippcca, vgnm`

48

pc

Purpose

This function estimates the pair correlation for a point process based on a single sample.

Synopsis

```
R = pc(X,dr)
R = pc(X,dr,rm)
[R,r] = pc(X, ...)
```

Description

`R = pc(X,dr)` calculates the pair correlation for the point process sample **`X`**, type **`uint8`**, such that the spatial domain is divided into annular rings of radial width **`dr`**.

`R = pc(X,dr,rm)` calculates the pair correlation with annular rings of radius up to rm.

`[R, r] = pc(X, ...)` returns the inner radius of all annular rings such that the pair correlation can be plotted as **`plot(r,R)`**.

Example

The following code creates a sample FM halftone pattern and then uses the **`pc`** function to estimate the corresponding pair correlation. The

49

results are then plotted versus the inner radius of each annular ring.

```
X=ones(128,128)*0.125;
H=halftone(X, 'rwerror');
[R, r]=pc(H, 0.5, 30);
plot(r, R);
```

References

D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic Geometry and Its Applications*, John Wiley and Sons, New York, 1987.

D. L. Lau, G. R. Arce, and N. C. Gallagher, Green-noise digital halftoning, *Proceedings of the IEEE*, 86(12):2424-2444, December 1998.

See Also

`ddf`

50

pp

Purpose

This function converts a binary halftone pattern to a point process sample.

Synopsis

```
Y = pp(X)
```

Description

`Y = pp(X)` converts the binary halftone image **`X`**, of type uint8 or double, to a point process sample **`Y`** of type uint8 (logical) such that minority pixels are represented by ones. The purpose of **`pp`** is to ensure the proper data type for functions that operate on point process samples.

Example

The following code illustrates the use of pp to convert a halftone image to a point process prior to estimating the reduced second moment measure.

```
X=rand(128,128)>0.25;
K=rsmm(pp(X), 30);
imagesc(K);
```


51

References

D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic Geometry and Its Applications*, John Wiley and Sons, New York, 1987.

D. L. Lau, G. R. Arce, and N. C. Gallagher, Green-noise digital halftoning, *Proceedings of the IEEE*, 86(12):2424-2444, December 1998.

See Also

`intnsty, rsmm, pc, anis`

52

psd2

Purpose

This function performs the 2-dimensional power spectral density estimation using the Bartlett s method of averaging periodograms.

Synopsis

```
Y = psd2(X,d)
Y = psd2(X,d,f)
Y = psd2(X,d,f,s)
[Y,c] = psd2(X,...)
```

Description

`Y = psd2(X,d)` estimates the power spectral density of the 2-D signal **`X`** using spatial domain windows of size **`d`** where **`d`** is the 2-by-1 vector **`[M N]'`** of the scalar quantity **`N`** (**`N`**-by-**`N`** windows).

`Y = psd2(X,d,f)` estimates the power spectral density of the 2-D signal **`X`** using spatial domain windows of size **`d`** transformed into spectral domain windows of size **`f`** where **`f`** is the 2-by-1 vector **`[P Q]'`** of the scalar quantity **`P`** (**`P`**-by-**`P`** windows). If not specified, the default value for **`f`** is **`d`**.

`Y = psd2(X,d,f,s)` extracts spatial domain windows from **`X`** such that two consecutive windows are defined by a spatial shift of **`s`**

53

pixels where **s** is either the vector **[R S]'** or the scalar value **R** (**R**-by-**R** shift). If not specified, the default value for **s** is **d**.

[Y,c] = psd2(X,...) returns the number of windows used to calculate the power spectrum estimate in **c**.

Example

The following code creates a sample FM halftone pattern and then uses the **psd2** function to estimate the corresponding 2-D power spectral density. The results are then plotted versus the inner radius of each spectral ring.

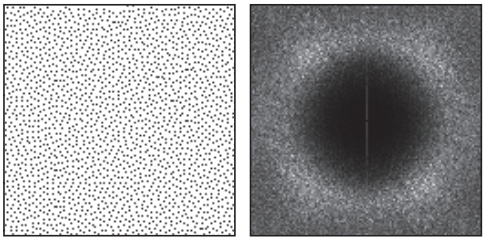
```
X=ones(128,1280)*0.125;
H=halftone(X, 'rwerror');
Y=psd2(double(H), 128, [], []);
Y(1)=0;  %delete the DC component
pcolor(fftshift(Y));
```

References

M. S. Bartlett, The spectral analysis of a point process, *Journal of the Royal Statistical Society*, *Series B*, 25(2):264-280, February 1964.

M. S. Bartlett, The spectral analysis of two-dimensional point processes, *Biometrika*, 51:299-311, December 1964.

54



A binary dither pattern and its corresponding spectral estimate produced by **psd2**.

See Also

rapsd, anis

55

rapsd

Purpose

This function estimates the radially averaged power spectrum density of a point process sample as the average power of the power spectrum density within a series of annular rings that partition the spectral domain.

Synopsis

```
P = rapsd(X,d)
P = rapsd(X,d,f)
P = rapsd(X,d,f,s)
[P,fr] = rapsd(X,...)
[P,fr,c]=rapsd(X,...)
```

Description

P = rapsd(X,d) calculates the Radially Averaged Power Spectrum Density metric, **P** of type **double**, from the sample image, **X** of type **double**, using periodograms of dimension **d** by **d**, where **d** is a scalar, with spectral rings of radial width equal to one pixel.

[P,fr] = rapsd(X,d) returns the inner radii of the spectral rings in the vector **fr** of type **double**.

[P,fr] = rapsd(X,d,f) calculates the power spectrum of **X**

56

using **d** by **d** windows transformed into **f** by **f** windows where **f** is a scalar. The parameter **f** may be empty or unspecified. In either case, **f** defaults to the value specified for **d**.

[P,fr] = rapsd(X,d,f,s) calculates the RAPSD metric where **s** is the amount of overlap between windows used to calculate the power spectrum estimate. The parameter **s** may be empty or unspecified. In either case, **s** defaults to the value 0.

[P,fr,c] = rapsd(X,...) returns the number of windows used to calculate the power spectrum estimate in **c**.

Example

The following code creates a sample FM halftone pattern and then uses the **rapsd** function to estimate the corresponding radially averaged power spectral density. The results are then plotted versus the inner radius of each spectral ring.

```
X=ones(128,1280)*0.125;
H=halftone(X, 'rwerror');
[P,fr]=rapsd(double(H), 128, [], []);
plot(fr,P);
```

References

M. S. Bartlett, The spectral analysis of a point process, *Journal of the Royal Statistical Society*, *Series B*, 25(2):264-280, February 1964.

57

M. S. Bartlett, The spectral analysis of two-dimensional point processes, *Biometrika*, 51:299-311, December 1964.

R. A. Ulichney, *Digital Halftoning*, MIT Press, Cambridge, MA, 1987.

See Also

`psd2, anis`

58

rgb2cmy

Purpose

This function converts a continuous-tone Red-Green-Blue image to a Cyan-Magenta-Yellow image.

Synopsis

```
Y = rgb2cmy(X)
```

Description

`Y = rgb2cmy(X)` converts a continuous-tone RGB image, of type *double* with intensity values ranging from 0 to 1, to a continuous-tone CMY image.

See Also

`rgb2cmyk, cmy2rgb, cmyk2rgb`

59

rgb2cmyk

Purpose

This function converts a continuous-tone Red-Green-Blue image to a Cyan-Magenta-Yellow-blacK image.

Synopsis

```
Y = rgb2cmyk(X)
```

Description

`Y = rgb2cmyk(X)` converts a continuous-tone RGB image, of type *double* with intensity values ranging from 0 to 1, to a continuous-tone CMYK image.

See Also

`rgb2cmy, cmy2rgb, cmyk2rgb`

60

rsmm

Purpose

This function calculates the reduced second moment measure given a sample of a point process.

Synopsis

```
K = rsmm(X)
K = rsmm(X,Rmx)
```

Description

`K = rsmm(X)` calculates the reduced second moment measure for the a point process given a sample **`X`**, type **`uint8`**`(`**`logical`**`)`.

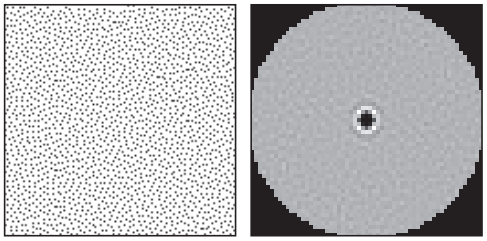
`K = rsmm(X,Rmx)` calculates the reduced second moment measure out to a maximum distance of **`Rmx`** pixels.

Example

The following code creates a sample FM halftone pattern and then uses the **`rsmm`** function to estimate the corresponding reduced second moment measure. The results are then shown using a pseudo-color plot.

```
X=ones(256,256)/8;
H=halftone(X, 'ulichney');
K=rsmm(H, 30);
imagesc(K);
```

61



A binary dither pattern and its corresponding reduced second moment measure estimate produced by **rsmm** where the maximum radius is set to 30 pixels. Areas in black, outside the 30 pixel radius, are undefined.

References

D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic Geometry and Its Applications*, John Wiley and Sons, New York, 1987.

D. L. Lau, G. R. Arce, and N. C. Gallagher, Green-noise digital halftoning, *Proceedings of the IEEE*, 86(12):2424-2444, December 1998.

62

See Also

`psd2`

63

ulichney

Purpose

This function converts a continuous-tone image to binary using Ulichney s perturbed error filter weight scheme.

Synopsis

```
H = ulichney(X)
```

Description

`H = ulichney(X)` converts the continuous-tone image **`X`**, type **`double`**, using Ulichney s perturbed error filter weight scheme.

Example

The following code converts a continuous-tone monochrome image into a blue-noise halftone using Ulichney s perturbed filter weight scheme.

```
load womanGRAY
H=ulichney(X);
imshow(H);
```

References

R. A. Ulichney, *Digital Halftoning*, MIT Press, Cambridge, MA, 1987.

64

The gray-scale image *woman* (left) before and (right) after halftoning with Ulichney s perturbed error filter weight scheme.

See Also

`errdif`

65

vpc

Purpose

This function estimates the pair correlation between channels or colors of a point process based on a single sample.

Synopsis

```
R = vpc(X,C,dr)
R = vpc(X,C,dr,rm)
[R,r] = vpc(X,...)
```

Description

`R = vpc(X,C,dr)` calculates the pair correlation between channels from **`C`**(1) to **`C`**(2) (**`C`** is a 2-by-1 vector) for the point process sample **`X`**, type **`uint8`**, such that the spatial domain is divided into annular rings of radial width **`dr`**.

`R = vpc(X,C,dr,rm)` calculates the pair correlation between colors with annular rings of radius up to **`rm`**.

`[R, r] = vpc(X, ...)` returns the inner radius of all annular rings such that the pair correlation can be plotted as **`plot(r,R)`**.

Example

The following code creates a sample FM halftone pattern and then uses the **`vpc`** function to estimate the corresponding pair correlation. The

66

results are then plotted versus the inner radius of each annular ring.

```
X=ones(128,128,2)*0.125;
H=zeros(size(X));
H(:,:,1)=halftone(X(:,:,1), 'rwerror');
H(:,:,2)=halftone(X(:,:,2), 'rwerror');
[R, r]=vpc(H, [1 2], 0.5, 30);
plot(r, R);
```

References

D. Stoyan, W. S. Kendall, and J. Mecke, *Stochastic Geometry and Its Applications*, John Wiley and Sons, New York, 1987.

D. L. Lau, G. R. Arce, and N. C. Gallagher, Digital color halftoning via generalized error-diffusion and multi-channel green-noise masks, *IEEE Transactions on Image Processing*, 9(5), May 2000.

See Also

`pc, rsmm`