anis	
bayer	
bippcca	
bippsma	
bnm	
cdod	
cmy2cmyk	
cmy2rgb	
cmyk2cmy	
cmyk2rgb	
ddf	
dotdif	
errdif	
gnm	
hlfmsk	
hvs	
imshowCMYK	
intnsty	
mbippcca	
pc	
pp	

2

3

function reference **Function Listing by Purpose** COLOR CONVERSIONS AND DISPLAY cmy2cmyk converts a continuous-tone CMY image to CMYK cmy2rgb...... converts a continuous-tone CMY image to RGB cmyk2cmy converts a continuous-tone CMYK image to CMY converts a continuous-tone CMYK image to RGB imshowCMYK displays CMY and CMYK images on screen rgb2cmy....... converts a continuous-tone RGB image to CMY rgb2cmyk...... converts a continuous-tone RGB image to CMYK HALFTONING ALGORITHMS performs Bayer s dither cdod performs clustered-dot ordered dithering dotdif performs dot diffusion errdif..... performs error-diffusion hlfmsk applies a halftone mask lau performs error-diffusion with adaptive hysteresis ... performs error-diffusion with perturbed weigths MASK ALGORITHMS bippcca ... the BInary Pattern Pair Correlation Construction Algorithm bippsma the Blnary Pattern Power Spectrum Matching Algorithm bnm the Blue-Noise Mask construction algorithm . the Green-Noise Mask construction algorithm

4

function reference **Function Listing by Chapter** 1.1 AM digital halftoningcdod 1.1.1 Screen Frequency 1.1.2 Dot Shape ... 1.1.3 Screen Angle 1.2 FM digital halftoningbayer 2 Halftone Statistics 2.3 Spectral Statisticspsd2, rapsd, anis 2.4 Halftone Visibility hvs lue-Noise Dithering 3 Blue-Noise Dithering 3.1 Spatial and Spectral Characteristics 3.2 Error Diffusion errdif 3.2.1 Eliminating Unwanted Textures ulichney, dotdiff, edodf 3.2.2 Edge Enhancement edshrp

6

Software for Modern Digital Halftoning
3.3 Direct Binary Search
4 Blue-Noise Masks
4.1 BIPPSMAbippsma,bnm
4.2 Simulated Annealingsimann
4.3 Void and Clustervac
5 Printers: Distortions and Models
5.1 Printer Distortion
5.1.1 Dot Gain
5.1.2 Dot Loss
5.2 Dot Models
5.2.1 Physical Modelsdgm
5.2.2 Statistical Models
5.3 Corrective Measures
5.3.1 Tone Correctiontcc
5.3.2 Model Based Halftoning mdlbsd
5.3.3 Clustering
6 Green-Noise Dithering
6.1 Spatial and Spectral Characteristics
6.1.1 Spatial Statistics
6.1.2 Spectral Statistics
6.2 Error-Diffusion with Output Dependent Feedback
6.2.1 Eliminating Unwanted Textures
6.2.2 Edge Enhancement
6

7

Green-Noise Masks 7.1 BIPPCCA	7.1 BIPPCCA 7.1.1 Pattern Robustness using BIPPCCA 7.1.2 Constructing the Green-Noise Mask 9n 7.2 Optimal Green-Noise Masks 9lor Printing 8.1 RGB, CMY and CMYK 8.1.1 RGB 8.1.2 CMY 8.1.2 CMY 8.2 Statistics 8.2 Statistics 8.3 Generalized Error Diffusion 8.4 Multi-channel Green-Noise Masks 8.4 L Color BIPPCCA 8.5 Indeed Support	7 Green-Noise Masks	7 Green-Noise Masks 7.1 BIPPCCA 7.1.1 Pattern Robustness using BIPPCCA 7.1.2 Constructing the Green-Noise Mask 9.2 Optimal Green-Noise Masks 8 Color Printing 8.1 RGB, CMY and CMYK 8.1.1 RGB 9.1.2 CMY 9.1.3 CMYK 9.1.3 CMYK 9.1.3 CMYK 9.2 Statistics 9.2 Statistics 9.3 Generalized Error Diffusion 9.4 Multi-channel Green-Noise Masks 9.4 Multi-channel Green-Noise Masks 9.5 Masks 9.6 Statistics 9.7 Statistics 9.8 Masks 9.8 Multi-channel Green-Noise Masks	function reference
7.1 BIPPCCA bippe 7.1.1 Pattern Robustness using BIPPCCA 7.1.2 Constructing the Green-Noise Mask g 7.2 Optimal Green-Noise Masks Color Printing 8.1 RGB, CMY and CMYK imshowCM 8.1.1 RGB rgb2cmy, rgb2cm 8.1.2 CMY cmy2rgb, cmy2cm 8.1.3 CMYK cmyk2rgb, cmyk2c 8.2 Statistics y 8.3 Generalized Error Diffusion gerrd	7.1 BIPPCCA 7.1.1 Pattern Robustness using BIPPCCA 7.1.2 Constructing the Green-Noise Mask 97.2 Optimal Green-Noise Masks 9100r Printing 8.1 RGB, CMY and CMYK 8.1.1 RGB 8.1.2 CMY 8.1.2 CMY 9.1.3 CMYK 8.1.2 CMY 8.1.3 CMYK 8.1.3 CMYK 8.1.4 Cmyk2rgb, cmyk2rgb 8.1.5 CMY 8.1.5 GMY cmyk2rgb, cmyk2rgb 8.2 Statistics 8.2 Statistics 98.3 Generalized Error Diffusion 98.4 Multi-channel Green-Noise Masks 98.4 L Color BIPPCCA 99.5 Masks 99.7 Masks 99.8 Malt-Color BIPPCCA 99.8 Masks 99.8 Malt-Color BIPPCCA 90.8 Masks 99.8 Malt-Color BIPPCCA 90.8 Masks 90.8 90.8 M	7.1 BIPPCCA bippec 7.1.1 Pattern Robustness using BIPPCCA 7.1.2 Constructing the Green-Noise Mask gr 7.2 Optimal Green-Noise Masks 8 Color Printing 8.1 RGB, CMY and CMYK imshowCMY 8.1.1 RGB rgb2cmy, rgb2cmy 8.1.2 CMY cmy2rgb, cmy2cmy 8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics rg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks 8.4.1 Color BIPPCCA mbippec	7.1 BIPPCCA. bippec 7.1.1 Pattern Robustness using BIPPCCA. 7.1.2 Constructing the Green-Noise Mask gr 7.2 Optimal Green-Noise Masks. 8 Color Printing. 8.1 RGB, CMY and CMYK imshowCMS 8.1.1 RGB. rgb2cmy, rgb2cmy 8.1.2 CMY cmy2rgb, cmy2cmy 8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics vg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mbippec 8.4.1 Color BIPPCCA mbippec	
7.1.1 Pattern Robustness using BIPPCCA	7.1.1 Pattern Robustness using BIPPCCA	7.1.1 Pattern Robustness using BIPPCCA	7.1.1 Pattern Robustness using BIPPCCA	vise Masks
7.1.2 Constructing the Green-Noise Mask	7.1.2 Constructing the Green-Noise Maskgr 7.2 Optimal Green-Noise Mask	7.1.2 Constructing the Green-Noise Maskgr 7.2 Optimal Green-Noise Masks	7.1.2 Constructing the Green-Noise Mask	PPCCAbippcc
7.2 Optimal Green-Noise Masks Color Printing 8.1 RGB, CMY and CMYK 8.1.1 RGB 8.1.2 CMY 8.1.3 CMYK 9.2 Statistics 9.3 Generalized Error Diffusion 9 gerrd 9 gerrd	7.2 Optimal Green-Noise Masks slor Printing	7.2 Optimal Green-Noise Masks 8 Color Printing I RGB imshowCMS 8.1 RGB rgb2cmy, rgb2cmy 8.1.2 CMY cmy2rgb, cmy2cmy 8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics vg 8.3 Generalized Error Diffusion gerraf 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbippcc	7.2 Optimal Green-Noise Masks 8 Color Printing	1.1 Pattern Robustness using BIPPCCA
Color Printing imshowCM 8.1 RGB, CMY and CMYK imshowCM 8.1.1 RGB rgb2cmy,rgb2cm 8.1.2 CMY cmy2rgb,cmy2cm 8.1.3 CMYK cmyk2rgb,cmyk2c 8.2 Statistics v 8.3 Generalized Error Diffusion gerrd	olor Printing imshowCM 8.1 RGB, CMY and CMYK imshowCM 8.1.1 RGB rgb2cmy, rgb2cmy 8.1.2 CMY cmy2rgb, cmy2cmy 8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics vg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbippcc	8 Color Printing imshowCMY 8.1 RGB, CMY and CMYK imshowCMY 8.1.1 RGB rgb2cmy rgb2cmy 8.1.2 CMY cmy2rgb, cmy2cmy 8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics vg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbippcc	8 Color Printing 8.1 RGB, CMY and CMYK imshowCMY 8.1.1 RGB rgb2cmy, rgb2cmy 8.1.2 CMY cmy2rgb, cmy2rgb, cmy2rgb 8.1.3 CMYK cmyk2rgb, cmyk2rg 8.2 Statistics vg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbippcc	1.2 Constructing the Green-Noise Maskgr
8.1 RGB, CMY and CMYK imshowCM 8.1.1 RGB rgb2cmy, rgb2cm 8.1.2 CMY cmy2rgb, cmy2cm 8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics v 8.3 Generalized Error Diffusion gerrd	8.1 RGB, CMY and CMYK 8.1.1 RGB 8.1.2 CMY 8.1.3 CMYK 8.1.3 CMYK 8.1.3 CMYK 8.1.3 Generalized Error Diffusion 8.4 Multi-channel Green-Noise Masks 8.4.1 Color BIPPCCA 8.5 ImmshowCM3 8.7 cmyk2gcmy, rgb2cmy 8.7 cmyk2rgb, cmyk2cm 9 cmyk2rgb, cmyk2rgb 9 cmy	8.1 RGB, CMY and CMYK imshowCM3 8.1.1 RGB rgb2cmy, rgb2cmy 8.1.2 CMY cmy2rgb, cmy2cmy 8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics vg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbippcc	8.1 RGB, CMY and CMYK imshowCM3 8.1.1 RGB rgb2cmy, rgb2cmy 8.1.2 CMY cmy2rgb, cmy2rgb, cmy2cmy 8.1.3 CMYK cmyk2rgb, cmyk2cmy 8.2 Statistics rgeratic geratic g	timal Green-Noise Masks
8.1 RGB, CMY and CMYK imshowCM 8.1.1 RGB rgb2cmy, rgb2cm 8.1.2 CMY cmy2rgb, cmy2cm 8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics v 8.3 Generalized Error Diffusion gerrd	8.1 RGB, CMY and CMYK imshowCMY 8.1.1 RGB rgb2cmy, rgb2cmy 8.1.2 CMY cmy2rgb, cmy2rgb, cmy2rgb 8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics vg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgn 8.4.1 Color BIPPCCA mbippcc	8.1 RGB, CMY and CMYK imshowCMY 8.1.1 RGB rgb2cmy, rgb2cmy 8.1.2 CMY cmy2rgb, cmy2cmy 8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics vg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgm 8.4.1 Color BIPPCCA mbippcc	8.1 RGB, CMY and CMYK imshowCM3 8.1.1 RGB rgb2cmy, rgb2cmy 8.1.2 CMY cmy2rgb, cmy2rgb, cmy2cmy 8.1.3 CMYK cmyk2rgb, cmyk2cmy 8.2 Statistics rgeratic geratic g	nting
8.1.2 CMY cmy2rgb, cmy2cm 8.1.3 CMYK cmyk2rgb, cmyk2c 8.2 Statistics v 8.3 Generalized Error Diffusion gerrd	8.1.2 CMY cmy2rgb, cmy2cmy 8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics vg 8.3 Generalized Error Diffusion gerddi 8.4 Multi-channel Green-Noise Masks mgn 8.4.1 Color BIPPCCA mbippec	8.1.2 CMY	8.1.2 CMY	
8.1.3 CMYK	8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics yg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbippec	8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics vg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbippcc	8.1.3 CMYK cmyk2rgb, cmyk2cm 8.2 Statistics vg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbippcc	1.1 RGBrgb2cmy,rgb2cmy
8.2 Statistics	8.2 Statistics vg. 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgn 8.4.1 Color BIPPCCA mbippec	8.2 Statistics vg 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgn 8.4.1 Color BIPPCCA mbippcc	8.2 Statistics VE 8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbippcc	1.2 CMYcmy2rgb,cmy2cmy
8.3 Generalized Error Diffusiongerrd	8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbbippec	8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbbippcc	8.3 Generalized Error Diffusion gerrdi 8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbippcc	1.3 CMYKcmyk2rgb, cmyk2cm
	8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbipped	8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbippec	8.4 Multi-channel Green-Noise Masks mgr 8.4.1 Color BIPPCCA mbipped	tistics vg
0.114.11.1.10.114.114.1	8.4.1 Color BIPPCCA mbippcc	8.4.1 Color BIPPCCAmbippcc	8.4.1 Color BIPPCCAmbippcc	neralized Error Diffusiongerrdi
8.4 Multi-channel Green-Noise Masks mg				ılti-channel Green-Noise Masksmgr
8.4.1 Color BIPPCCA mbippc	onclusions	9 Conclusions	O Conclusions	4.1 Color BIPPCCA mbippcc
Conclusions				ons

8

Software for Modern Digital Halftoning

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

 $\label{eq:Asigma} \textbf{A} = \texttt{anis}(\textbf{X}, \textbf{d}) \ \text{calculates the Anisotropy metric}, \textbf{A} \ \text{of type } \textbf{double}, \\ \text{from the sample image}, \textbf{X} \ \text{of type } \textbf{double}, \\ \text{using periodograms of } \text{dimension } \textbf{d} \ \text{by } \textbf{d}, \\ \text{where } \textbf{d} \ \text{is a scalar}, \\ \text{with spectral rings of radial } \\ \text{width equal to one pixel}.$

 $[\mathtt{A},\mathtt{fr}] = \mathtt{anis}(\mathtt{X},\mathtt{d})$ returns the inner radii of the spectral rings in the vector \mathtt{fr} of type \mathtt{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,

function reference

 ${\tt f}$ defaults to the value specified for ${\tt d}$.

[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.

9

10

Software for Modern Digital Halftoning

See Also

ddf, pc, rapsd

10

11

function reference

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

Software for Modern Digital Halftoning





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

June 11-13, 1973.

See Also

hlfmsk, cdod

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 = bippca(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.

 $\mathtt{H} = \mathtt{bippca}(\mathtt{N}, \mathtt{g}, \mathtt{asoc}, \mathtt{X})$ uses \mathtt{X} as the initial or starting dither pattern such that \mathtt{X} is a subset of the resulting pattern \mathtt{H} . \mathtt{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.

13

14

Software for Modern Digital Halftoning

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);
```

Reference

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

14

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

 ${\tt H} = {\tt bippsma}\,({\tt N}, {\tt g})\,$ generates a blue-noise binary dither pattern of size ${\tt N}$ -by- ${\tt N}\,({\tt N}\,$ scalar) representing gray level ${\tt g}.$

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

 ${\tt H} = {\tt bippsma}({\tt N}, {\tt g}, {\tt w}, {\tt seed})$ uses ${\tt seed}$ 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

Software for Modern Digital Halftoning

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, $\it Journal of Electronic Imaging, 3(1):92-97, January$

1994. See Also bippsma function reference

17

18

Software for Modern Digital Halftoning

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 *iih* 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

15

19

function reference

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

Software for Modern Digital Halftoning

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 $\mathtt{size}(X,3)$ 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;



function reference

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);

21

function reference

23

22

Software for Modern Digital Halftoning

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

22

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

Software for Modern Digital Halftoning

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

function reference

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

25

26

Software for Modern Digital Halftoning

cmyk2rgb

Purpos

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

26

27

function reference

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

Software for Modern Digital Halftoning

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

function reference

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

 $\mathbf{H} = \mathbf{dotdif}(\mathbf{X}, \mathbf{M})$ converts the continuous-tone image \mathbf{X} into the binary halftone image \mathbf{H} using Knuth s dot diffusion. Pixels, within a halftone cell, are processed in the sorted order of pixels in the class matrix, \mathbf{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));
```

29

30

Software for Modern Digital Halftoning





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

References

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

31

function reference

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

See Also

errdif, hlfmsk

32

Software for Modern Digital Halftoning

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.

Synopsi

```
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 \mathbf{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];
```

32

function reference

where **b** represents an error filter coefficient and **a** represents a hysteresis filter coefficient. The parameter ${\bf k}$ 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 ${\tt P}$ and the amount of perturbation is specified by ${\tt T}$. The matrices ${\tt P}$ and ${\tt T}$ are the same size as Wand are of the form of:

```
P=[ 1 2 1 3 4
3 5 4 5 6
                    T=[1 1 1 1 1
                      11111
   6 2 0 7 8
                       1 1 0 1 1
   9 8 10 9 10
                       11111
      7 12 11 12];
                       1 1 1 1 1]*0.25;
  11
```

such that weights with equal values in ${\bf P}$ are paired and the amount of perturbation is equal to the corresponding values in T times the smaller of the two weights.

33

34

Software for Modern Digital Halftoning

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, ÚSA, 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

35



function reference

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

Software for Modern Digital Halftoning

gnm

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

Synopsis

Y = qnm(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 ${\bf G}$ with an average cluster size of asoc. G also specifies the order to which patterns are constructed such the ith constructed pattern represents gray-level G(i) with clusters of average size $\mathbf{asoc}(i)$ pixels. The first two values in $\mathbf G$ must be 0and 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

function reference

(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

37

38

Software for Modern Digital Halftoning

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 ${\bf X}$ is a color image, ${\bf M}$ 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

function reference

The CMYK image woman (left) before and (right) after halftoning using a four chan-

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

Software for Modern Digital Halftoning

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

function reference

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

 $\mathbf{Y} = \mathbf{hvs}(\mathbf{X}, \mathbf{v}, \mathbf{d})$ returns the apparent image as defined by a model of the human visual system with a viewing distance of \mathbf{v} (in inches) and a print resolution of \mathbf{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

41

function reference

42

Software for Modern Digital Halftoning

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.

42

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

Software for Modern Digital Halftoning

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.

function reference

See Also

pp, rsmm, pc, anis

45

46

Software for Modern Digital Halftoning

mbippcca

Purpos

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

```
\begin{array}{lll} \mathtt{BP} &= & \mathtt{vbippcca}\left(\mathtt{H}, \mathtt{g}, \mathtt{r}, \mathtt{R1}, \mathtt{R2}, \ldots, \mathtt{Rc}, \mathtt{F1}, \mathtt{F2}, \ldots, \mathtt{Fc}\right) \\ \mathtt{BP} &= & & \mathtt{vbippcca}\left(\mathtt{H}, \mathtt{g}, \mathtt{r}, \mathtt{R1}, \ldots, \mathtt{Rc}, \mathtt{F1}, \ldots, \mathtt{Fc}, \mathtt{M}\right) \end{array}
```

Description

BP = vbippeca (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 = vbippeca(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.

4

47

function reference

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

Software for Modern Digital Halftoning

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

48

function reference

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

49

50

Software for Modern Digital Halftoning

pp

Purpose

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

Synopsi

Y = pp(X)

Description

 $\mathbf{Y} = \mathbf{pp}(\mathbf{X})$ converts the binary halftone image \mathbf{X} , of type uint8 or double, to a point process sample \mathbf{Y} of type uint8 (logical) such that minority pixels are represented by ones. The purpose of \mathbf{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);
```

50

51

function reference

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

Software for Modern Digital Halftoning

psd2

Purpos

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] 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

function reference

pixels where ${\tt s}$ is either the vector [R S] ${\tt r}$ or the scalar value R (R-by-R shift). If not specified, the default value for ${\tt s}$ is ${\tt d}$.

[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.

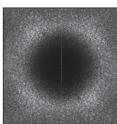
M. S. Bartlett, The spectral analysis of two-dimensional point processes, *Biometrika*, 51:299-311, December 1964.

53

54

Software for Modern Digital Halftoning





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

See Also

rapsd, anis

54

55

function reference

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.

 $\begin{tabular}{ll} $[P,fr] = rapsd(X,d)$ returns the inner radii of the spectral rings in the vector fr of type $double. \end{tabular}$

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

- --

56

Software for Modern Digital Halftoning

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

.

function reference

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

 $R.\,A.\,Ulichney, \textit{Digital Halftoning}, MIT\,Press, Cambridge, MA, 1987.$

See Also

psd2, anis

57

58

Software for Modern Digital Halftoning

rgb2cmy

Purpos

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

58

59

function reference

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

Software for Modern Digital Halftoning

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

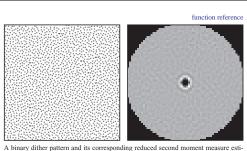
K = rsmm(X) calculates the reduced second moment measure for the a point process given a sample X, type uint8 (logical).

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);
```



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

61

62

Software for Modern Digital Halftoning See Also psd2

63

function reference

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 ${\tt double}, using \ Ulichney \ s \ perturbed \ error \ filter \ weight \ scheme.$

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);

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

function reference

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 uint 8, 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 ${\bf vpc}$ function to estimate the corresponding pair correlation. The

66

Software for Modern Digital Halftoning

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
```