Gernot Hoffmann
Dithering + Halftoning

Floyd-Steinberg

Hilbert-Peano
Floyd-Steinberg
Raster Halftoning
References

Contents

Settings for Acrobat

Edit / Preferences / General / Page Display (since version 6)
Custom Resolution 72 dpi

Edit / Preferences / General / Color Management (full version only)
sRGB
EuroscaleCoated or ISOCoated or SWOP
GrayGamma 2.2

1.1 Dithering by Hilbert-Peano

Based on an idea of Peano, David Hilbert had published 1891 the mathematical description of a function, which maps all points of a plane to the points of a line. This curve has some features which are very helpful for the dithering of images.
- The curve walks always in her own vicinity
- The curve hits all points (dithering dots) in the plane exactly once
- The curve seems to walk on a random path

Along the path all color values, e.g. cyan, are accumulated. If the sum is greater than 255, a cyan point is printed and in the accumulator 255 is subtracted. This is a bilevel printing: cyan or nothing. Multilevel printing with different dotsizes is possible as well. The curve can be programmed in powers of 2. Here we have the width 256.
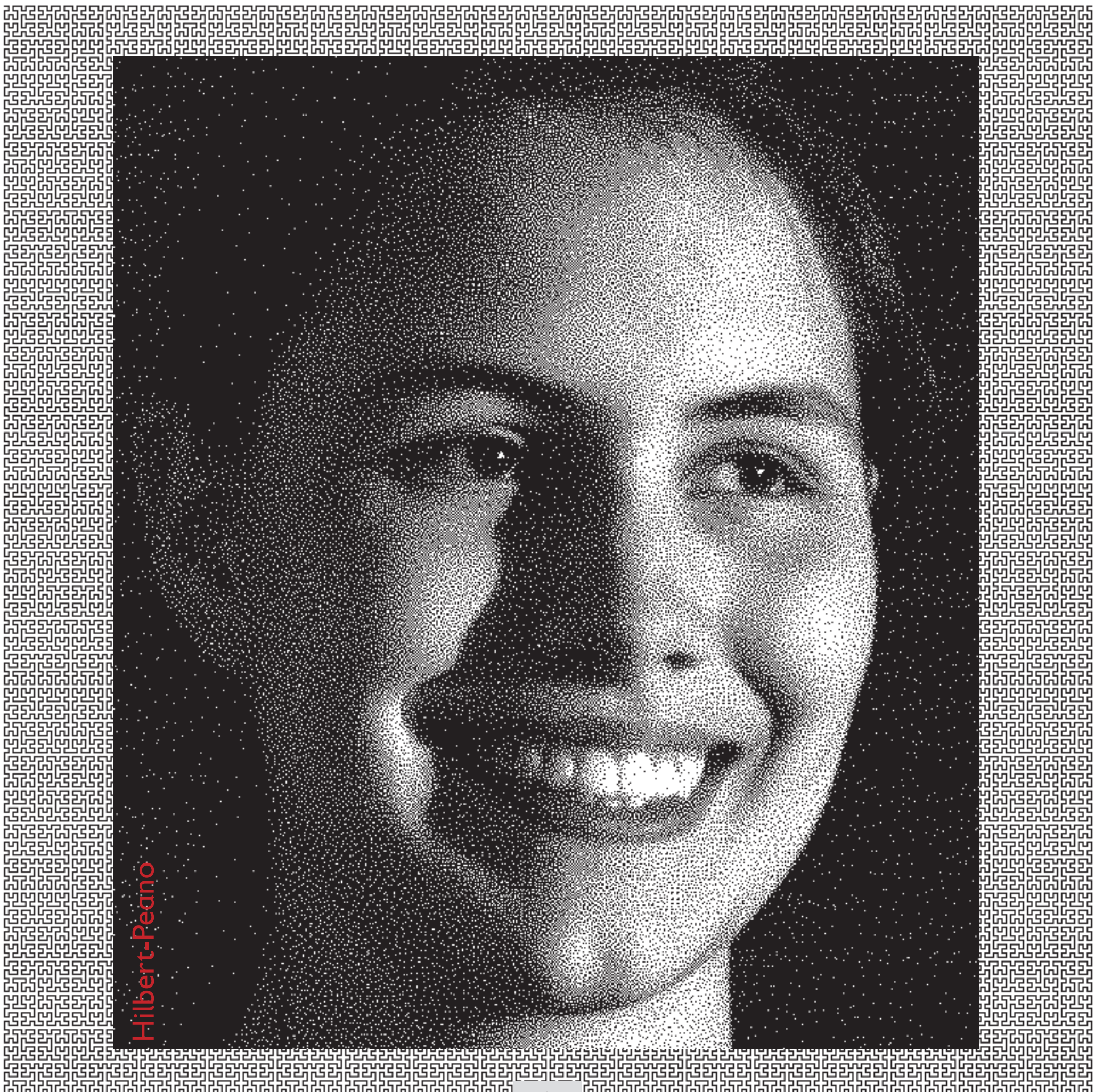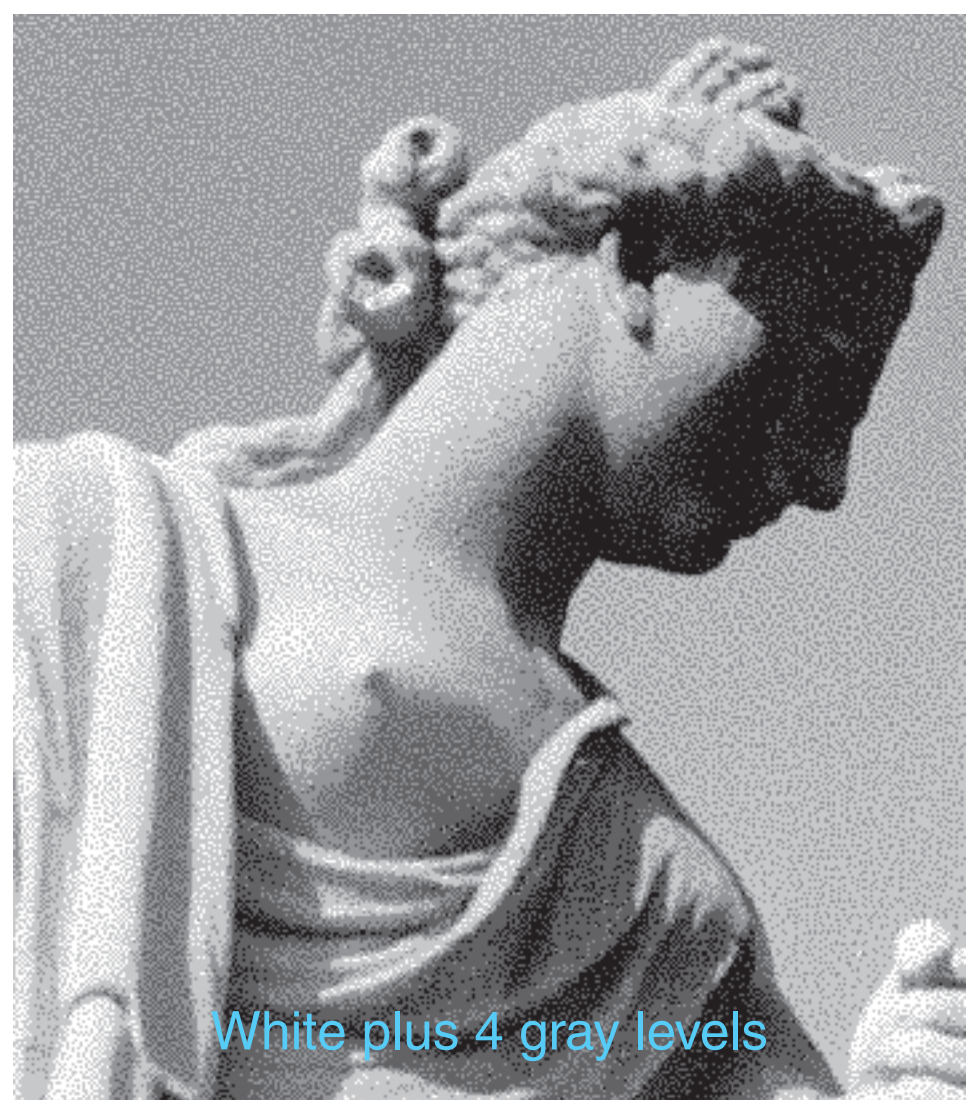

Hilbert-Peano

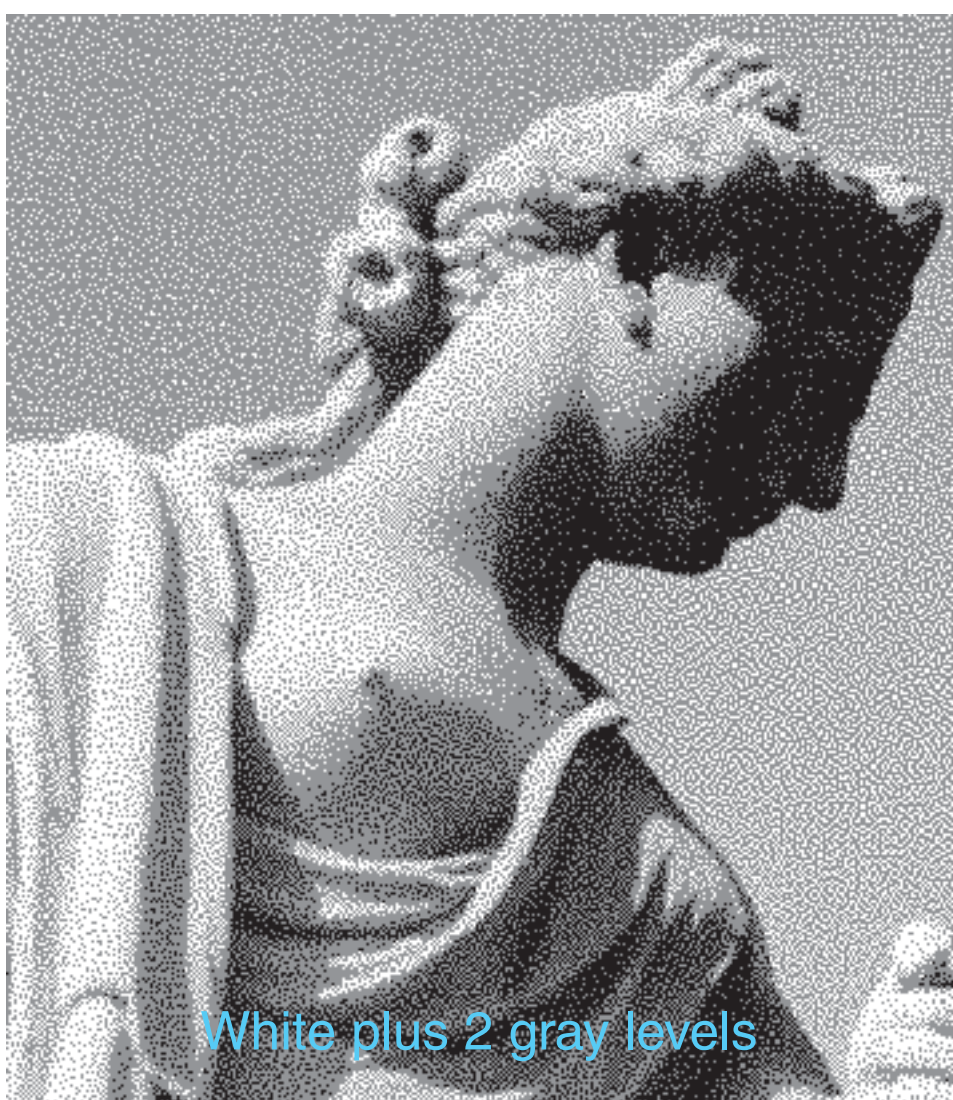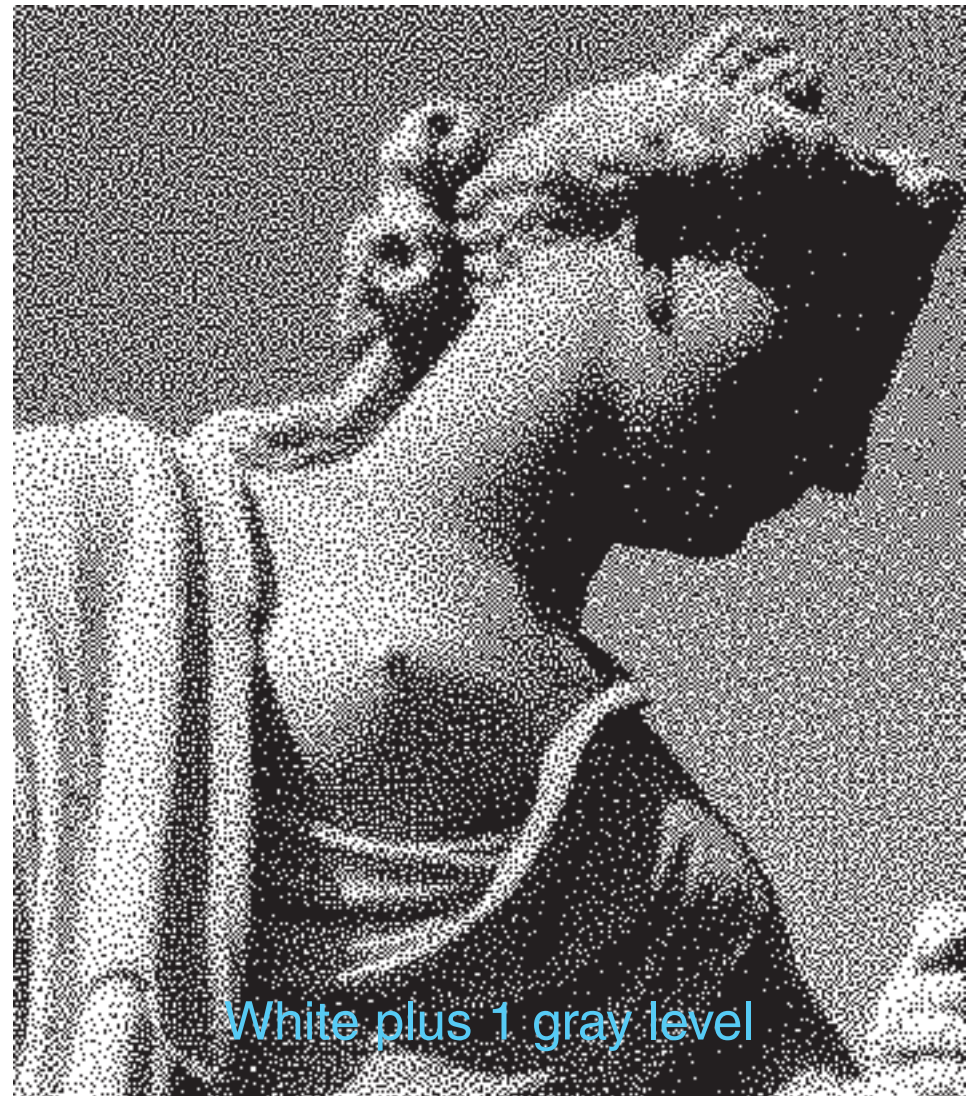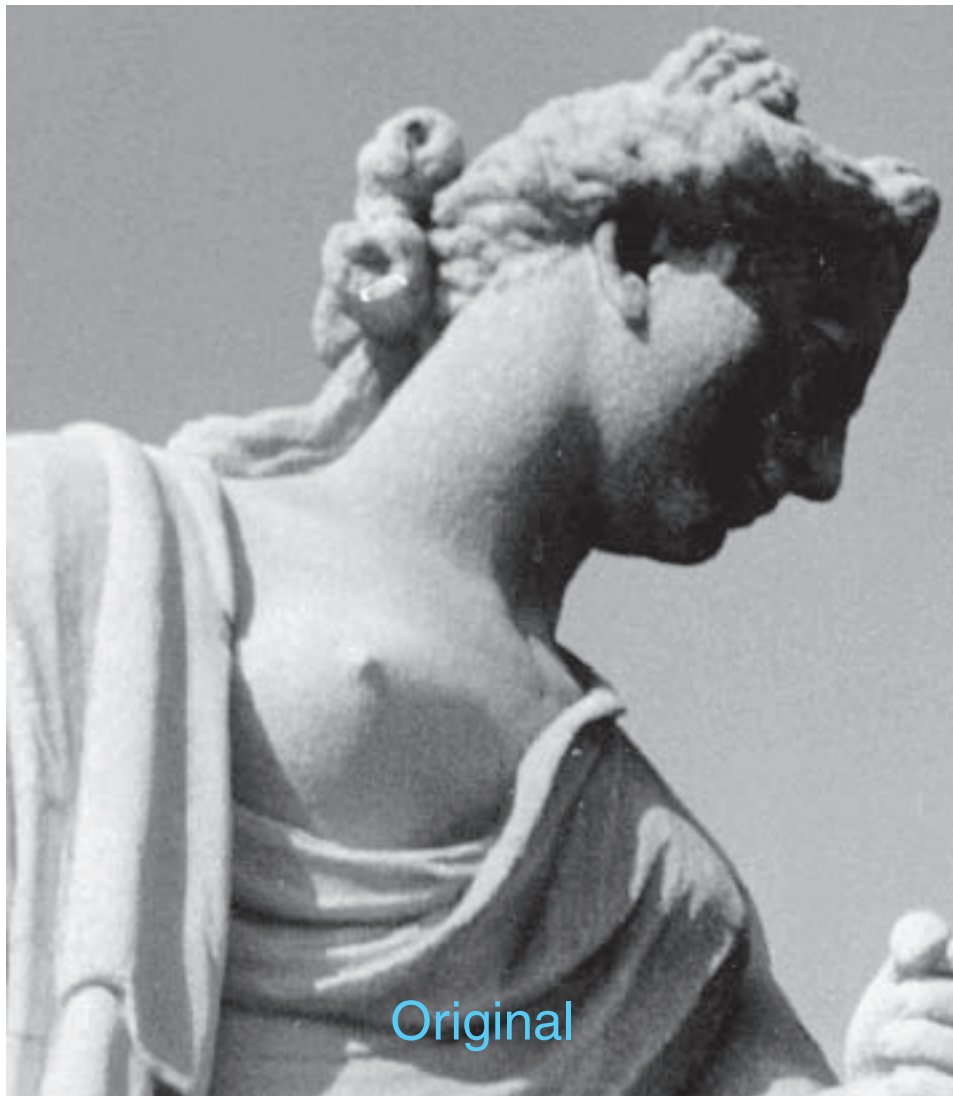# 1.2 Dithering by Hilbert-Peano

This page shows an original Grayscale Image and Hilbert-Peano Dithering with one, two and four gray levels.
This is a screen simulation. Instead of small printer dots we have large pixels.
Image Processing by ZEBRA.

For correct pixel view download and use Acrobat Reader Zoom 100% or 200%

Original

White plus 1 gray level

White plus 2 gray levels

White plus 4 gray levels

2.1 Dithering by Floyd-Steinberg

This page shows an original Grayscale Image and Floyd-Steinberg Dithering with one, two and four gray levels.
This is a screen simulation. Instead of small printer dots we have large pixels.
Image Processing by ZEBRA.

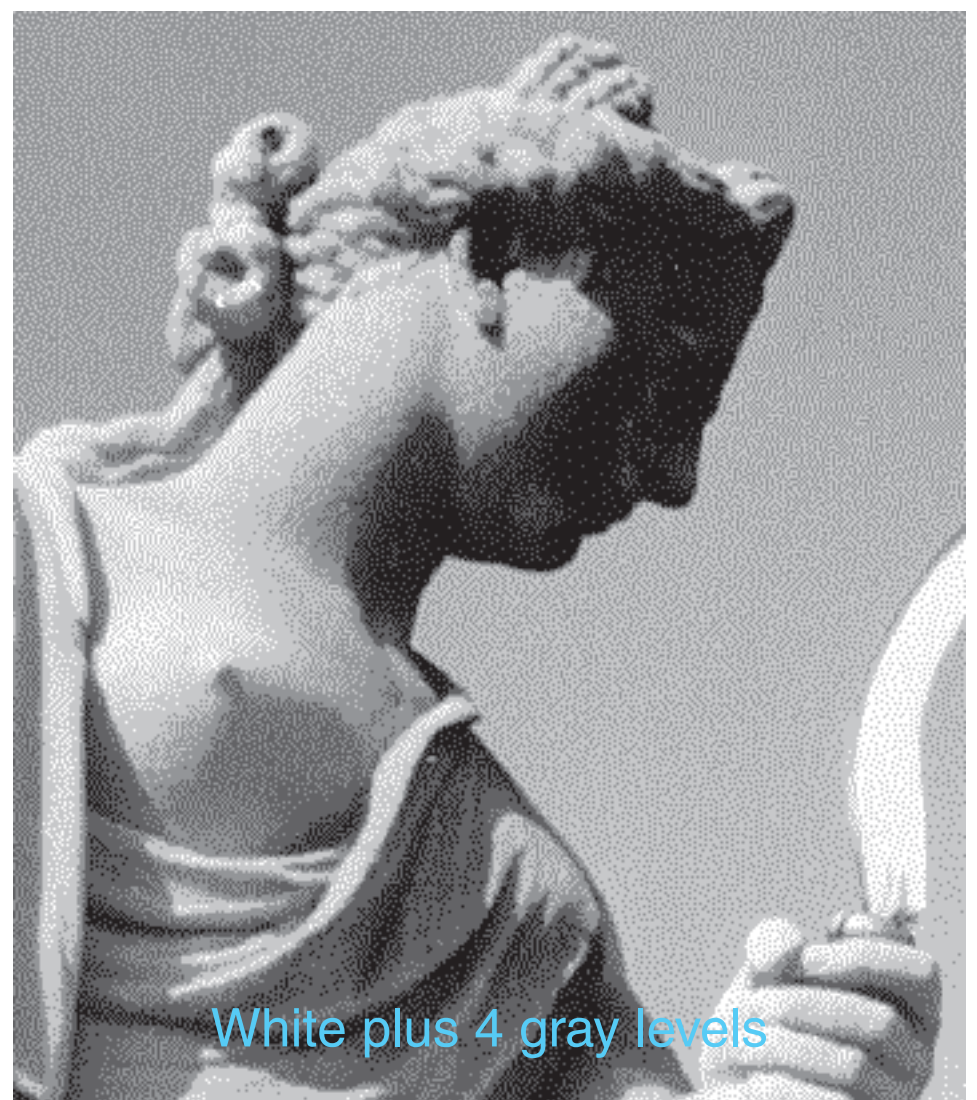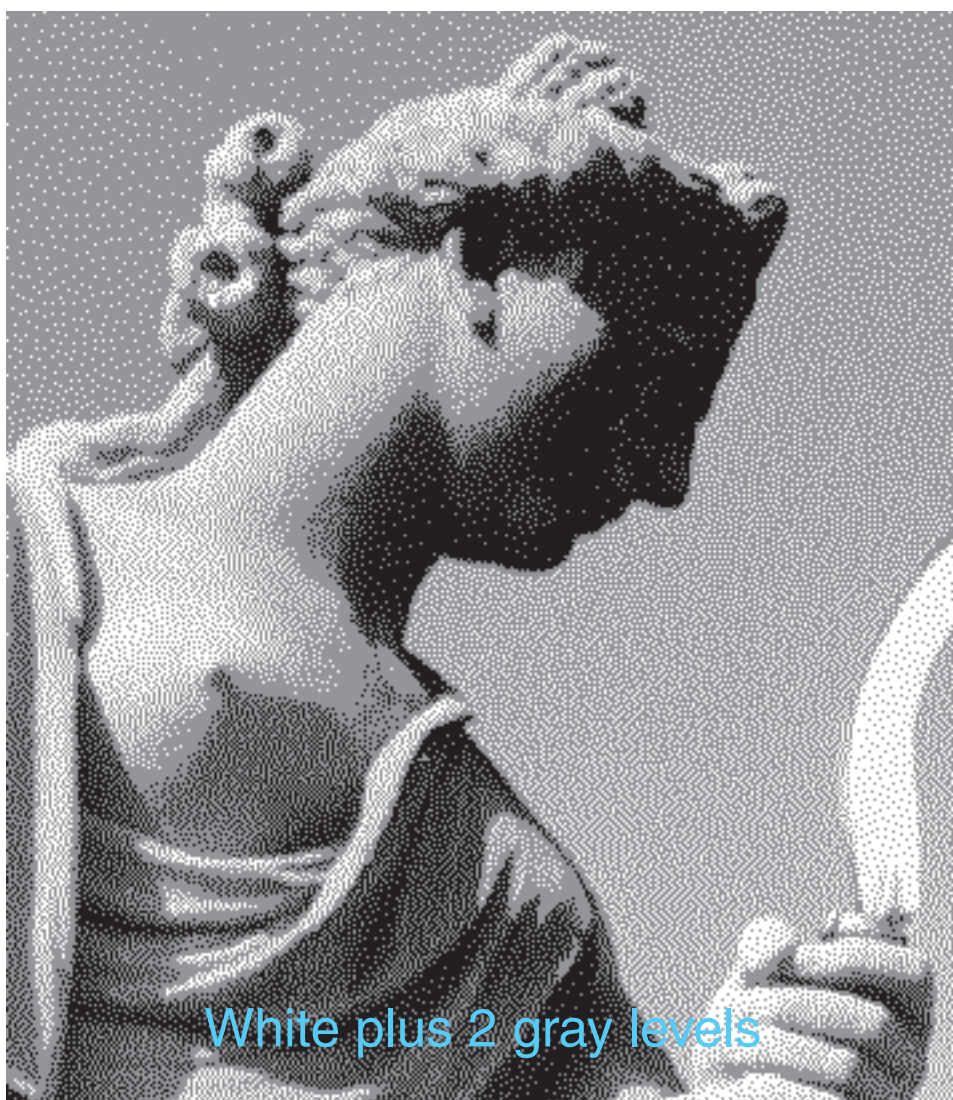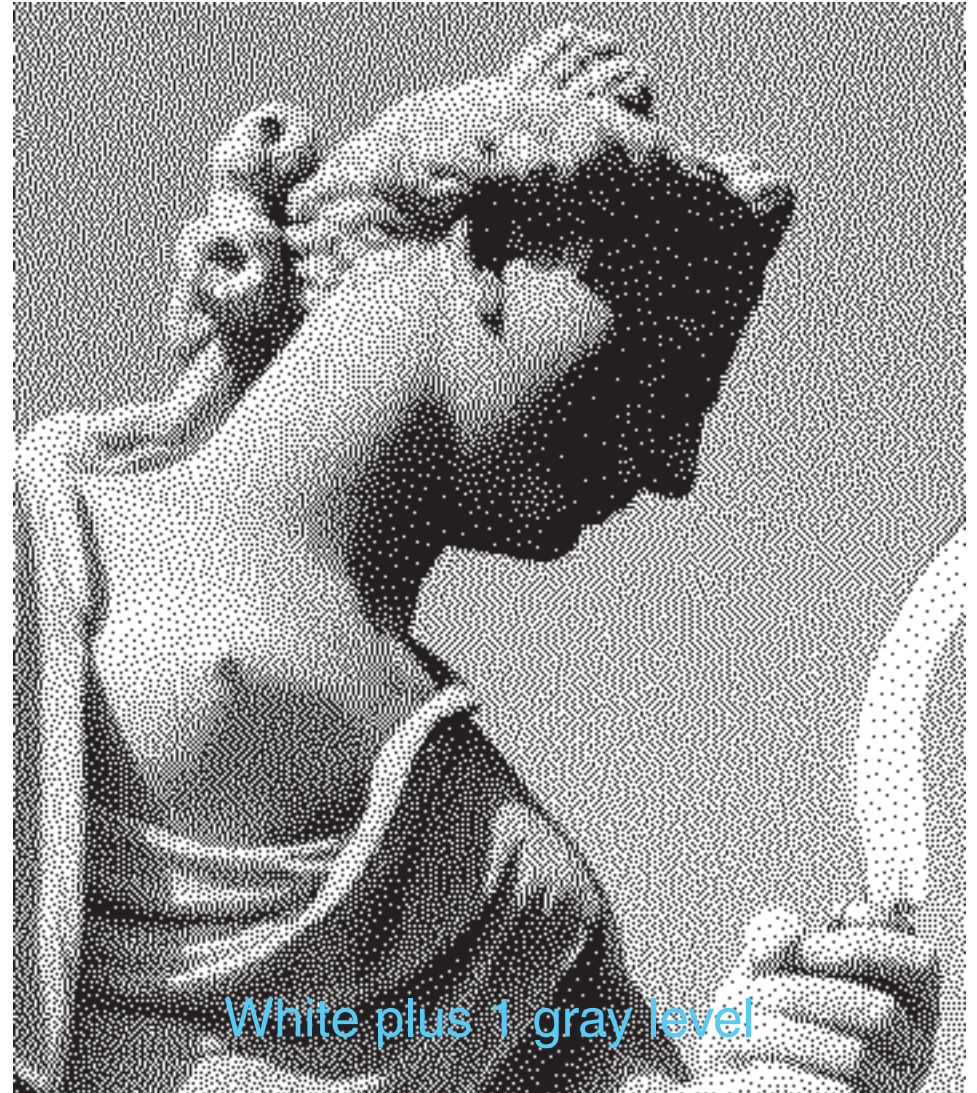For correct pixel view download and use Acrobat Reader Zoom 100% or 200%

Original (previous page)

White plus 1 gray level

White plus 2 gray levels

White plus 4 gray levels

2.2 Dithering by Floyd-Steinberg

The Floyd-Steinberg algorithm was programmed as recommended in [3] with alternating directions for even and odd rows. This is the error weighting scheme for left to right dithering in even rows:

y=0	x=0										xmx
y	x	7/16			x	7/16					x
	5/16	1/16		3/16	5/16	1/16				3/16	5/16
ymx											

Bilevel Dithering: here each destination color C=R,G,B is either 0 or 255. Hilbert-Peano (page 3) delivers generally less sharp results, compared to Floyd-Steinberg (page 1, cover). Artifacts may result in synthetical images in uniform or shaded areas, but nearly never in photos. The Hilbert-Peano algorithm is based on [1] and [2].

Floyd-Steinberg creates also artifacts in synthetical images. Sometimes it is helpful to dither first trilevel and then bilevel.

For monitor simulations of bilevel dithering it is generally necessary to cancel the working space gamma to some extent. Recalculate all source colors C=R,G,B by $C=C^G$. Gamma is not necessarily 2.2. Sometimes it´s better to use 1.6.

If images with bilevel dithering or halftoning are converted to smooth images by blurring filters then it´s necessary to apply an inverse gamma correction by $C=C^{1/2.2}$ for 2.2 working spaces. This has nothing to do with the gamma correction in advance to the dithering or halftoning.

2.3 Dithering by Floyd-Steinberg
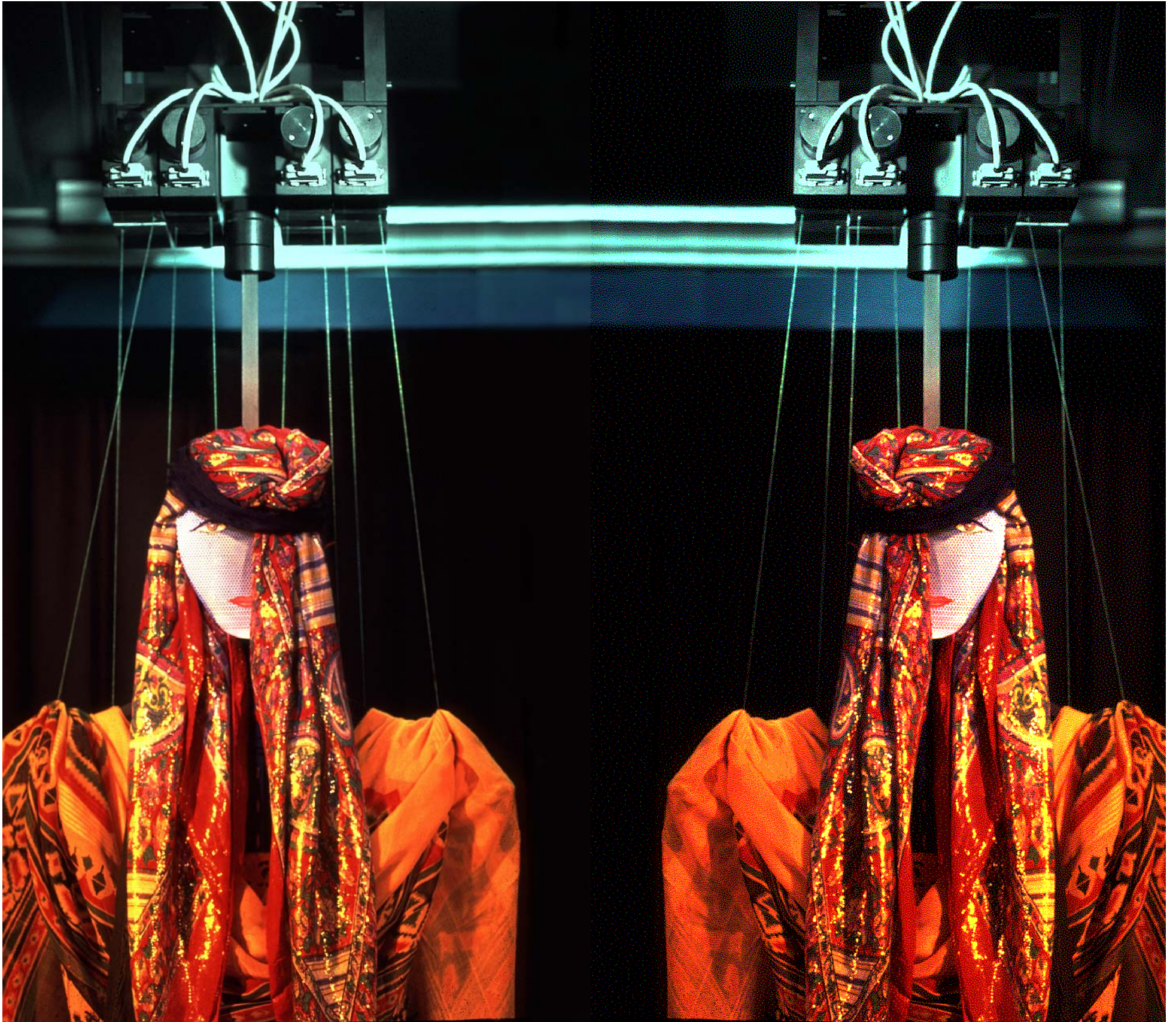
For correct pixel view download and use Acrobat Reader Zoom 200%

The Floyd-Steinberg algorithm can be used for color quantization.

Left: original image.

Right: three levels for each channel R,G,B .

3.1 Raster Halftoning

This is a tutorial demonstration of raster halftoning. It´s used for offset or laser printing, either for gray images or for each color channel separately.

Each raster cell contains one spot. A spot consists of printable dots with device resolution. We don´t call these dots pixels. Raster cells are arranged in Lpi distance, dots in dpi distance. A square raster cell with n dots in one direction can reproduce $m=n^2$ levels plus white.

The image is virtually placed on the paper. A frame with raster cell width is placed on the paper. For each device dot the underlying source image pixel value is taken and compared with a spot function.

In a previous version of this doc it was assumed that the gray values in the frames are averaged. This assumption was wrong.

We are starting with a simplified explanation: The gray in a uniform area defines the height for the spot function. Everything inside the height contour has to be filled by black.

Light gray would deliver a nearly round spot. 50% gray a square spot and dark gray four white quarter circles at the corners which appear as round white spots on the printed paper. Black and white dots are complementary to each other with respect to medium gray 50%.
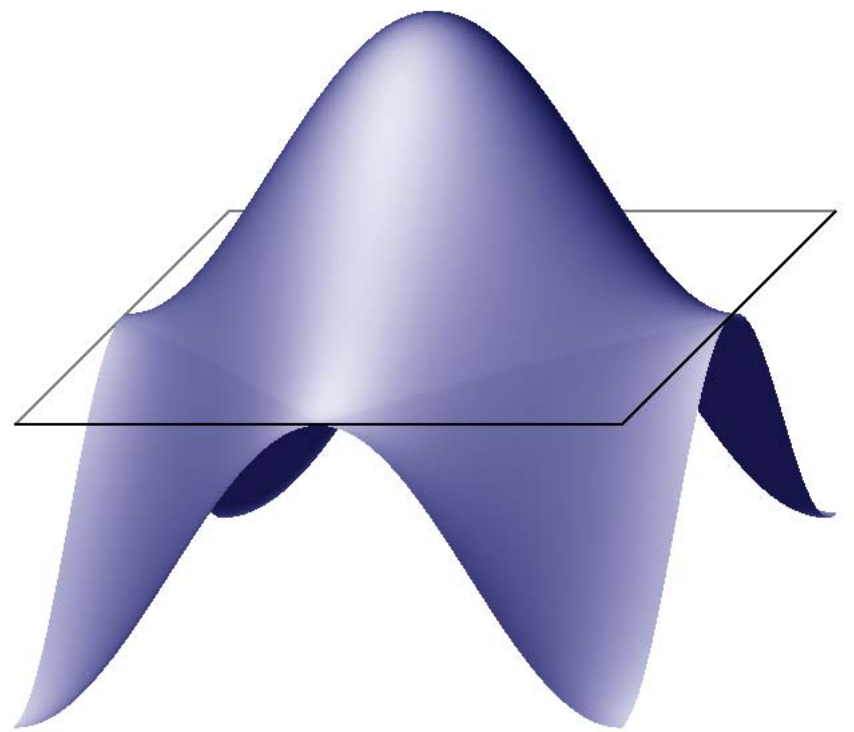
Actually it works different. Quoted by [4], where we have replaced 'pixel' by 'dot':
'The values the spot function returns are not significant. All that matters is the *relative* spot function values for different dots. A cell´s gray value varies from black to white, the first dot whitened is the one whose spot function has the lowest value, the next pixel is the one with the next higher spot function value, and so on. If two dots have the same spot function value, **setscreen** chooses their relative order arbitrarily.'

One may think that each cell has *one* gray. As already mentioned this is wrong. Nothing is averaged - the underlying image pixels are directly used, which preserves sharp image elements but may cause fragments of spots.

The spot function can have any shape and height, it´s by no means necessary to normalize the function, let´s say for height one. Of course the spot function is defined for one raster cell.

The mechanism becomes better understandable by the introduction of threshold arrays.

3.2 Raster Halftoning

The image shows a threshold array for n=6. A dot is drawn black if the generating pixel delivers a gray value below the threshold. The threshold arrays are here created by spot functions. A float increment is calculated by $dg=1/m$ with $m=n^2$. The first decision level is $1-dg/2$. The other levels are generated by decrementing. The lowest decision level is $dg/2$. The levels are multiplied by 255 and rounded.

```
C:\ZESNP\ZESNP001.BMP
36 Levels     Increment 7.083
    25   81  166  152   67   18
    53  110  223  216  103   46
   138  195  251  244  181  131
   124  188  237  230  174  117
    39   96  209  202   89   32
     4   74  159  145   60   11
```

The threshold array is filled by zeros. The raster cell is scanned on a spiral contour which starts in the center for *odd* n or with a half step offset left and up for *even* n. This is the strategy: find all values of the spot function which are not yet used and which are not larger than the previous value. Then find the maximum of these. This delivers a new entry for the array. Decrement the level. Execute the scan m times. For n=16 the lowest threshold would be zero. This has to be replaced by 1, because dots are drawn black if the gray value is *below* the level.
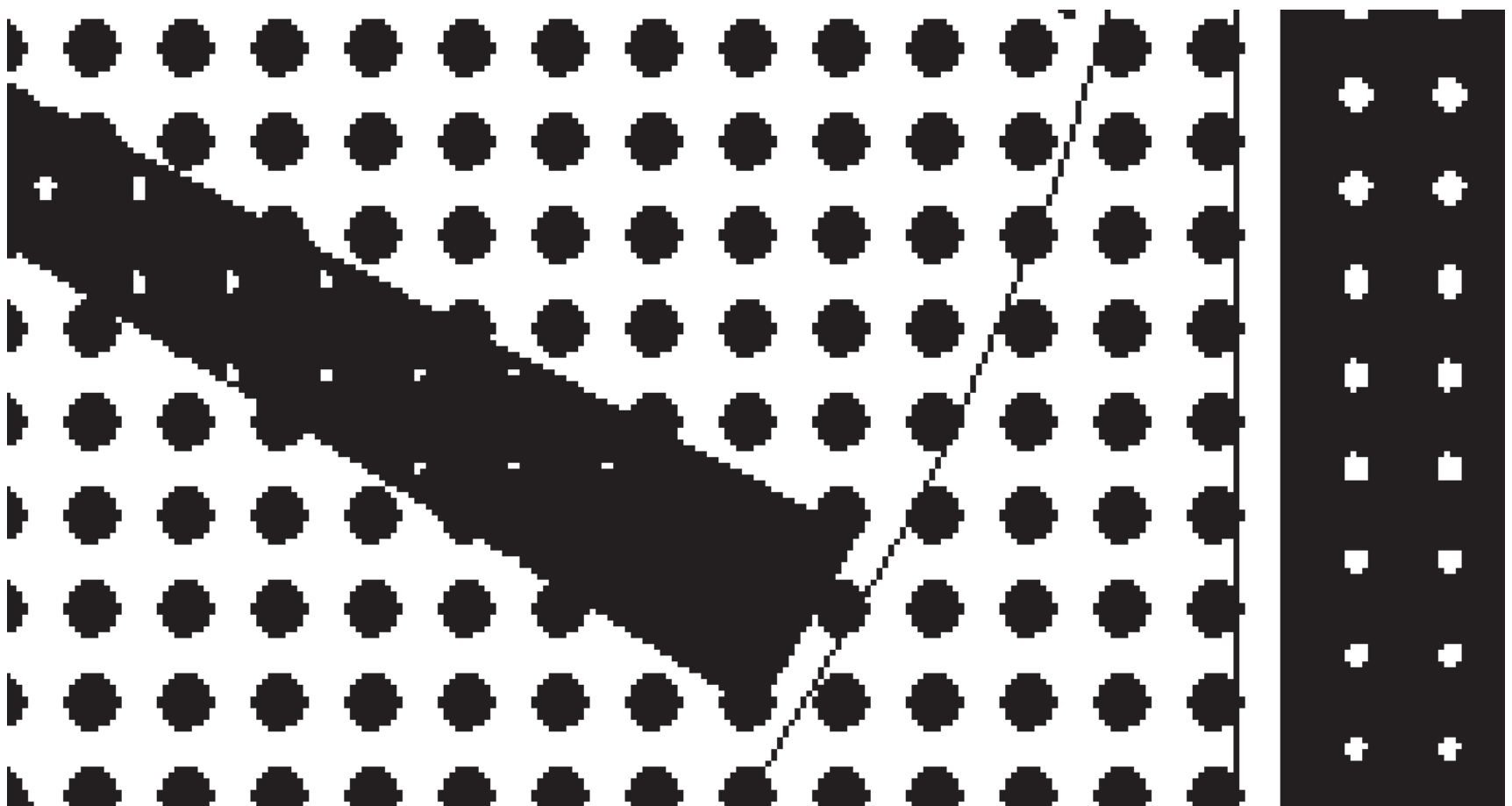
The image shows the rasterization for n=16 in a synthetical graphic

Screens with non-zero angle are created by a brute force method: rotate the image, rasterize the whole area and rotate back using nearest neighbour 'interpolation'. This works reasonably for 45° and for larger n. Better methods are described in [5].

In the example we have 16 pixels per cell width. The fuzzyness of the spots could be reduced by blurring the source image a little. This holds true mainly for photos.

3.3 Raster Halftoning Example

n=6, 36 levels, angle 0°
Use zoom 200%

3.4 Raster Halftoning Example

C:\ZESNP\ZESNP006.BMP
256 Levels   Increment 0.996

3	11	23	43	71	99	135	151	149	133	97	69	41	22	10	2
6	15	31	51	75	103	143	159	157	141	101	73	49	30	14	5
18	26	39	59	91	131	167	179	177	165	129	89	57	38	25	17
35	47	55	79	123	175	187	203	202	186	174	121	78	54	46	34
63	67	83	115	163	199	211	223	222	210	198	161	113	82	66	62
87	95	111	155	191	215	231	239	238	230	213	189	153	110	94	86
107	119	147	183	207	227	243	251	250	242	225	205	181	146	118	106
127	139	171	195	218	234	246	255	254	245	233	217	193	170	138	126
125	137	169	194	219	235	247	252	253	244	232	216	192	168	136	124
105	117	145	182	206	226	241	248	249	240	224	204	180	144	116	104
85	93	109	154	190	214	228	237	236	229	212	188	152	108	92	84
61	65	81	114	162	197	209	220	221	208	196	160	112	80	64	60
33	45	53	77	122	173	185	201	200	184	172	120	76	52	44	32
19	27	36	58	90	130	166	178	176	164	128	88	56	37	24	16
7	12	28	50	74	102	142	158	156	140	100	72	48	29	13	4
1	8	20	42	70	98	134	150	148	132	96	68	40	21	9	1

Zebra Sub Menue
K:\\GIESE012.BMP BMP-8
Raster Halftone
↑↓ 16.00 11.31 Pixel
D   Dot
S   Square
H   Horiz-Line
V   Verti-Line
E   Experiment
Q/A Angle = 45
N/M Gamma = 2.20
Δ∇   Threshold M.= Show
<   Undo Raster
ESC Menue
X   Shift Menue Area

n=16, 256 levels, angle 45°
Use zoom 200%

4.1 Code Floyd-Steinberg

```
Procedure IFloyd(ncol: Integer);
{   Copyright Gernot Hoffmann; November 20, 2001
    ncol: Color levels: white + 1,2,4,8,16,32,64,128
    Floyd-Steinberg
    Source FImage; Destination FImage; works inplace
    Uses global FImage=FMem[y]^[x]; packed prgb=pb/rb/gb/bb
              gmx : for x=0..gmx  Pixels, gmx<=1279
              gmy : for y=0..gmy  Pixels
    This is an edited program text. Errors cannot be excluded }

Var x,y,i,j,xa,ya,xe,ye,flap    :   Integer;
    pb                          :   Byte;
    ri,gi,bi,er,k1,k2,k3,c1,c2  :   Word;
    q,z,gam                     :   Single;
    r1,r2,g1,g2,b1,b2   : Array[-1..1280] Of Word;
    tabr,tabg,tabb      : Array[ 0.. 255] Of Byte;
Begin
xa:=0; ya:=0; xe:=gmx; ye:=gmy;
gam:=1.0;
Case ncol Of
 1: gam:=1.60;
 2: gam:=1.15;
End; { case }
For i:=0 to 255 Do
Begin q:=i/255;
XPowerA(q,gam,z     ,flap); Tabr[i]:=Round(255*z);
XPowerA(q,gam+0.05,z,flap); Tabg[i]:=Round(255*z);
XPowerA(q,gam,z     ,flap); Tabb[i]:=Round(255*z);
End;
Case ncol Of
1: Begin c1:=256;   c2:=8; End; {  1 color  }
2: Begin c1:=128;   c2:=7; End; {  2 colors }
3: Begin c1:= 64;   c2:=6; End; {  4 colors }
4: Begin c1:= 32;   c2:=5; End; {  8 colors }
5: Begin c1:= 16;   c2:=4; End; { 16 colors }
6: Begin c1:=  8;   c2:=3; End; { 32 colors }
7: Begin c1:=  4;   c2:=2; End; { 64 colors }
8: Begin c1:=  2;   c2:=1; End; {128 colors }
End;
For x:=xa to xe Do
Begin r1[x]:=0; r2[x]:=0; g1[x]:=0; g2[x]:=0; b1[x]:=0; b2[x]:=0;
End;
y:=ya;
Repeat
  For x:=xa to xe Do
    Begin
    GetFixel(x,y,pb,ri,gi,bi);

    er:=r1[x]+Tabr[ri];        er:=g1[x]+Tabg[gi];        er:=b1[x]+Tabb[bi];
    ri:=(er Div c1)*c1;        gi:=(er Div c1)*c1;        bi:=(er Div c1)*c1;
    If ri>255 Then ri:=255;    If gi>255 Then gi:=255;    If bi>255 Then bi:=255;
    er:=er-ri;                 er:=er-gi;                 er:=er-bi;
    k1:=(7*er)Div 16;          k1:=(7*er)Div 16;          k1:=(7*er)Div 16;
    k2:=(3*er)Div 16;          k2:=(3*er)Div 16;          k2:=(3*er)Div 16;
    k3:=(5*er)Div 16;          k3:=(5*er)Div 16;          k3:=(5*er)Div 16;
    r1[x+1]:=r1[x+1]+k1;       g1[x+1]:=g1[x+1]+k1;       b1[x+1]:=b1[x+1]+k1;
    r2[x-1]:=r2[x-1]+k2;       g2[x-1]:=g2[x-1]+k2;       b2[x-1]:=b2[x-1]+k2;
    r2[x  ]:=r2[x  ]+k3;       g2[x  ]:=g2[x  ]+k3;       b2[x  ]:=b2[x  ]+k3;
    r2[x+1]:=r2[x+1]+er-k1-k2-k3;  g2[x+1]:=g2[x+1]+er-k1-k2-k3;  b2[x+1]:=b2[x+1]+er-k1-k2-k3;

    SetFixel(x,y,pb,ri,gi,bi);
    End;
  For i:=xa to xe Do
    Begin
    r1[i]:=r2[i]; g1[i]:=g2[i]; b1[i]:=b2[i]; r2[i]:=0; g2[i]:=0; b2[i]:=0;
    End;
```

```
Inc(y); If y>ye Then Exit;

 For x:=xe DownTo xa Do
  Begin                            13
   GetFixel(x,y,pb,ri,gi,bi);

   er:=r1[x]+Tabr[ri];       er:=g1[x]+Tabg[gi];       er:=b1[x]+Tabb[bi];
   ri:=(er Div c1)*c1;       gi:=(er Div c1)*c1;       bi:=(er Div c1)*c1;
   If ri>255 Then ri:=255;   If gi>255 Then gi:=255;   If bi>255 Then bi:=255;
   er:=er-ri;                er:=er-gi;                er:=er-bi;
   k1:=(7*er)Div 16;         k1:=(7*er)Div 16;         k1:=(7*er)Div 16;
   k2:=(3*er)Div 16;         k2:=(3*er)Div 16;         k2:=(3*er)Div 16;
   k3:=(5*er)Div 16;         k3:=(5*er)Div 16;         k3:=(5*er)Div 16;
   r1[x-1]:=r1[x-1]+k1;      g1[x-1]:=g1[x-1]+k1;      b1[x-1]:=b1[x-1]+k1;
   r2[x+1]:=r2[x+1]+k2;      g2[x+1]:=g2[x+1]+k2;      b2[x+1]:=b2[x+1]+k2;
   r2[x  ]:=r2[x  ]+k3;      g2[x  ]:=g2[x  ]+k3;      b2[x  ]:=b2[x  ]+k3;
   r2[x-1]:=r2[x-1]+er-k1-k2-k3; g2[x-1]:=g2[x-1]+er-k1-k2-k3;  b2[x-1]:=b2[x-1]+er-k1-k2-k3;


   SetFixel(x,y,pb,ri,gi,bi);
  End;
 For i:=xa to xe Do
  Begin
  r1[i]:=r2[i]; g1[i]:=g2[i]; b1[i]:=b2[i]; r2[i]:=0; g2[i]:=0; b2[i]:=0;
  End;
 Inc(y);
 Until y>ye;
End; { IFloyd }
```

5.1 Code Raster Halftoning

```pascal
Procedure ISpotT(rwid,dot,alf,tmat: Integer; gam: Single);
{ G.Hoffmann
  January 07, 2004
  Convert Color to Black on White
  Rasterize by Threshold Matrix
  FMem    Works inplace in global framebuffer FMem
  rwid    Width of halftone cell in pixels, 2..16
  dot=1   Round
  dot=2   Square ...  }
Var x,y,i,j,k,p,m0,m1,mq,si,sj    : Integer;
    xa1,ya1,xe1,ye1,xm1,ym1,xi,yj : Integer;
    flag,n,kmax                   : Integer;
    xa4,ya4,xe4,ye4               : Integer;
    pp,mm,so,sn,sm,gry,dgr,x1,y1  : Single;
    prgb,pbbb,pwww                : LongInt;
    pb,rb,gb,bb                   : Byte;
    Pout                          : Boolean;
    txt1,txt2                     : String;
    Tab    : Array [0..255]    Of Byte;
    Thr    : Array [0..15,0..15] Of Byte;
    Procedure Dots;
    Begin
    If Not Pout And (Thr[i,j]=0) Then
     Begin
      { Function has maximum in center of raster cell
        or at the edges of the raster cell            }
        Case dot Of
         1: sn:=  coc(pp*i-pi)+0.90*coc(pp*j-pi); { Dot, coc=Cosine }
         2: sn:=-(Abs(i-mm)+0.90*Abs(j-mm));       { Square }
         3: sn:= -Abs(j-mm)-0.15*Abs(i-mm);        { x-Line }
         4: sn:= -Abs(i-mm)-0.15*Abs(j-mm);        { y-Line }
         5: sn:=-(Sqr(i-mm)+0.90*Sqr(j-mm));       { Circle }
        End; { Case}
        If sn<=so Then
        Begin
        If sn>sm Then
         Begin
          si:=i; sj:=j; sm:=sn;
         End;
        End;
     End;
    End;
Begin
{ Gamma correction gam=1.4..2.2 for 2.3 working space }
For i:=0 to 255 Do
Begin XPowerA(i/255,gam,gry,flag); Tab[i]:=Round(255*gry);
End;
{ Define white and black in 4-byte coding }
pwww:=255 SHL 16 + 255 SHL 8 +255;  { white 0 255 255 255 }
pbbb:=  1;                           { black 1   0   0   0 }
If rwid< 2 Then rwid:= 2;
If rwid>16 Then rwid:=16;
m0:=rwid-1;
m1:=rwid;              { width of cell                    }
mq:=Sqr(m1);          { number of levels, without white }
kmax:=mq;
mm:=0.5*m0;           { center, zero based               }
dgr:=1/mq;            { Increment in range 0..1          }
pp:=pi*2/m0;
pwww:=255 SHL 16 + 255 SHL 8 +255;  { white }
pbbb:=  1 SHL 24;                    { black }
{ Build threshold array }
For j:=0 to m0 Do
For i:=0 to m0 Do Thr[i,j]:=0;
```

5.2 Code Raster Halftoning

```
so:=+1E6;
gry:=1-dgr/2;
For k:=1 to kmax Do
Begin
{ Spiral scan }
sm:=-1E6;
pout:=False;
If odd(m1) Then i:=m0 Div 2 Else i:=m1 Div 2 -1;
j:=i; Dots;
n:=1;
 Repeat
  For p:=1 to n Do Begin Inc(i); If i>m0 Then Pout:=True; Dots; End;
  For p:=1 to n Do Begin Inc(j); Dots; End;
  Inc(n);
  For p:=1 to n Do Begin Dec(i); If i<0  Then Pout:=True; Dots; End;
  For p:=1 to n Do Begin Dec(j); Dots; End;
  Inc(n);
 Until Pout Or (n>m1);
 Thr[si,sj]:=Round(255*gry);
 If Thr[si,sj]=0 Then Thr[si,sj]:=1;
 so:=sm;
 gry:=gry-dgr;
End;
{ Find actual image size in frame buffer }
FrameLim ('F',xa1,ya1,xe1,ye1,xm1,ym1);
{ Gray Copy
  1=1/3 weight; 2=NTSC; 3=PhS            }
FMemGrYIQ (xa1,ya1,xe1,ye1,xa1,ya1,2);
ImiToScr; { Copy frame buffer to screen  }
{ Rotate image in frame buffer. Procedure is not in document }
RolTranP (+1,alf,rwid,xa1,ya1,xe1,ye1,xa4,ya4,xe4,ye4);
{ Rasterize }
y:=ya4;
Repeat
 x:=xa4;
 Repeat
 For j:=0 to m0 Do
 Begin
 yj:=y+j;
 For i:=0 to m0 Do
  Begin
  xi:=x+i;
 { Use blue channel of gray image via gamma table }
   bb:=FMem[yj]^[xi] AND $000000FF;
   If Tab[bb]<Thr[i,j] Then FMem[yj]^[xi]:=pbbb
                       Else FMem[yj]^[xi]:=pwww;
  End;
 End;
 x:=x+m1;
 Until x>xe4;
y:=y+m1;
Until y>ye4;
If (xa1<>0) Or (xe1<>gmx) Or (ya1<>0) Or (ye1<>gmy) Then
ColToImi(frab,whit); { Fill total area by white }
{ Rotate frame buffer back }
RolTranP(-1,alf,rwid,xa1,ya1,xe1,ye1,xa4,ya4,xe4,ye4);
End;
```

6.1 References

[1] Tesuo Asano
     Digital Halftoning Algorithm based on Random Space Filling Curve
     IEEE Int. Conf. on Image Proc., Lausanne 1996, Vol.1 p.545-548

[2] Sei-ichiro Kamata
     An Address Generator of an N-dimensional Hilbert Scan
     IEEE Int. Conf. on Image Proc., Lausanne 1996, Vol.2 p.1031-1034

[3] J.D.Foley+A.vanDam+St.K.Feiner+J.F.Hughes
     Computer Graphics
     Addison-Wesley Publishing Company, Reading, Massachusetts ..., 1993

[4] PostScript Language Reference Manual
     Addison-Wesley Publishing Company, Reading, Massachusetts ..., 1998

[5] Henry R.Kang
     Color Technology for Electronic Imaging Devices
     SPIE Optical Engineering Press, Bellingham, Washington USA, 1997

[6] About Color Rendering
     http://www.fho-emden.de/~hoffmann/colrend290800.pdf
     http://www.fho-emden.de/~hoffmann/spot290800.pdf

Gernot Hoffmann
September 24 / 2009 (January 01 / 2001)
Website
Load browser and click here