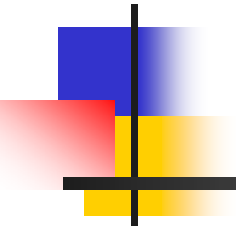
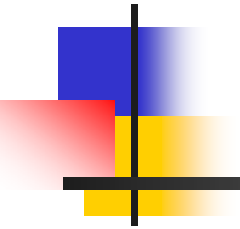
Digital Image Processing



Special topic:

Image Editing



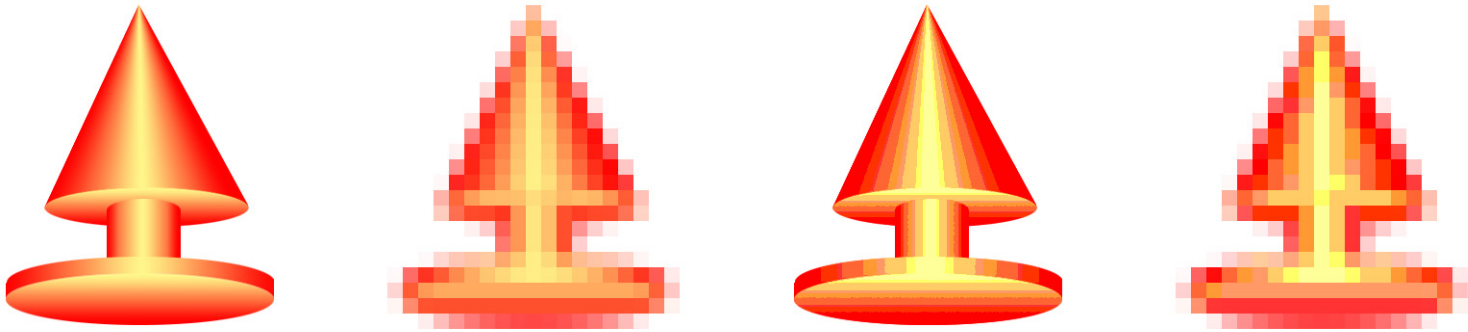
Lecture Notes: Pixelization and Quantization

Pixelization …

… is a special effect often used to hide identities …

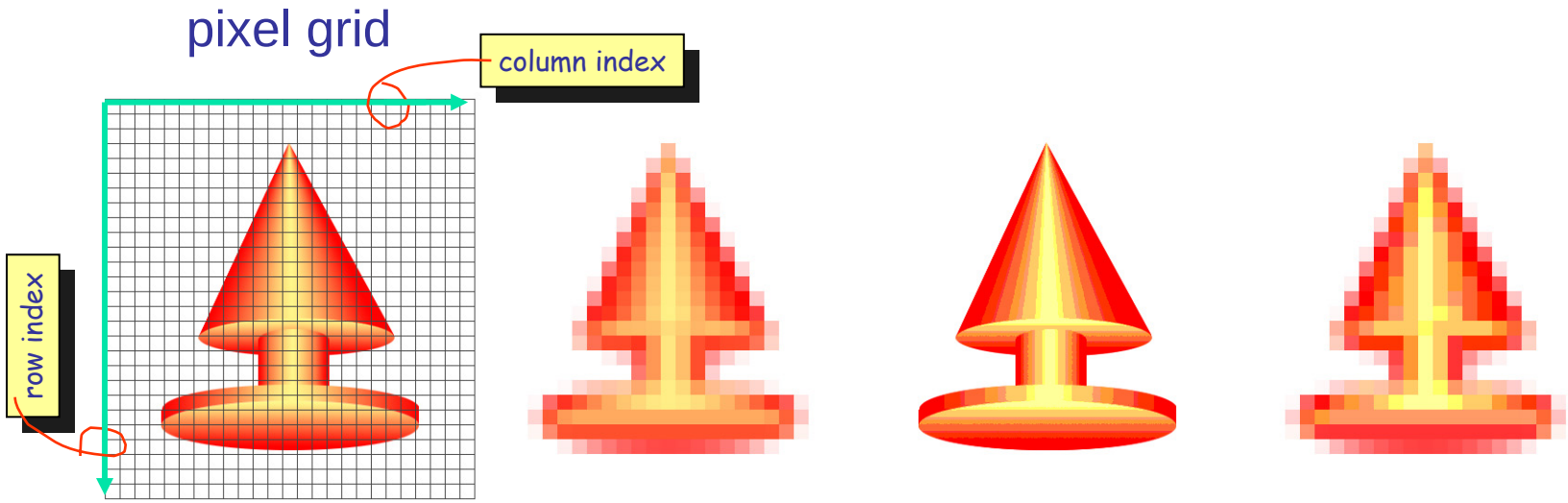
How to do it?

Pixelization and Quantization



high-res image | pixelated | quantized | pixelated & quantized

Pixelization and Quantization



high-res image

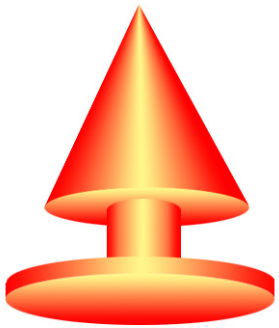
pixelated

quantized

pixelated
& quantized

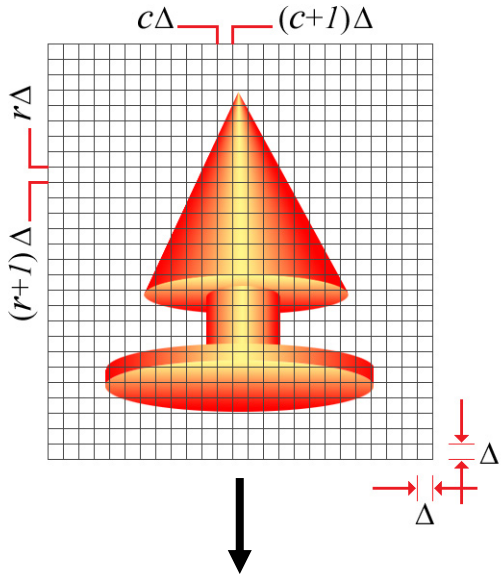
Pixelization

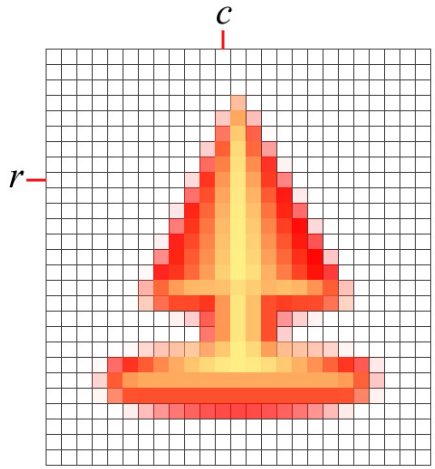
Take the average within each square.

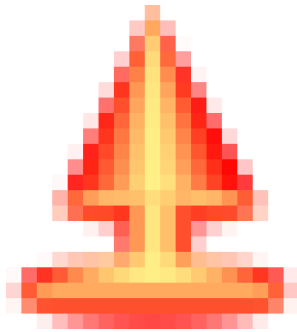


$I(\rho,\chi)$

high-res image







$I_P(r,c)$

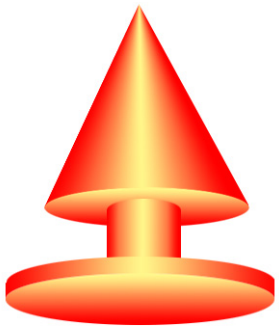
pixelated image

$$I_{DS}(r,c)=\frac{1}{\Delta^2}\sum_{\rho=r\Delta}^{(r+1)\Delta-1}\sum_{\chi=c\Delta}^{(c+1)\Delta-1} I(\rho,\chi); \quad I_P = I_{DS}\uparrow\Delta$$

(*upsampled*)

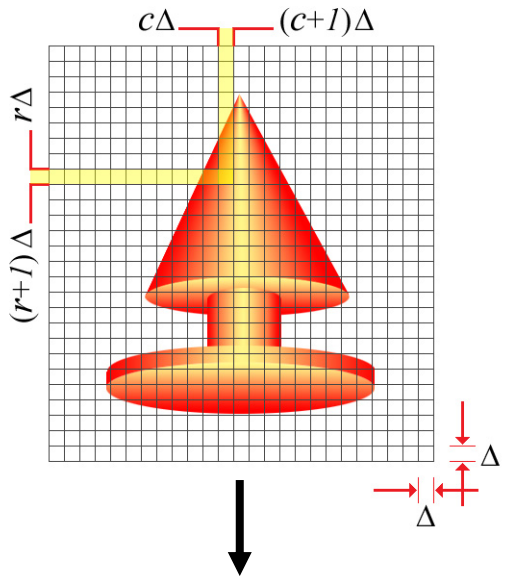
Pixelization

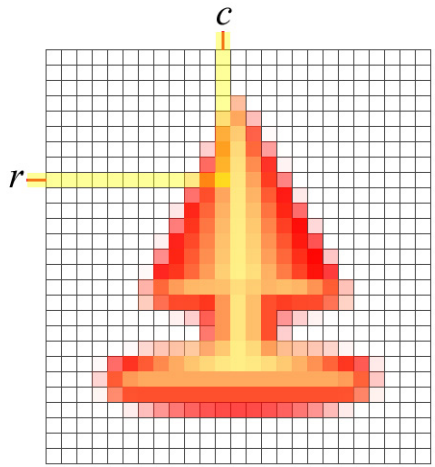
Take the average within each square.

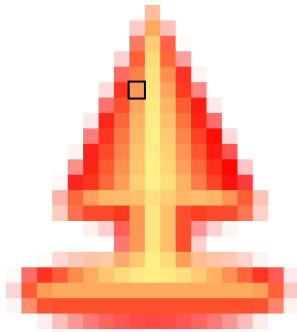


$I(\rho,\chi)$

high-res image







$I_P(r,c)$

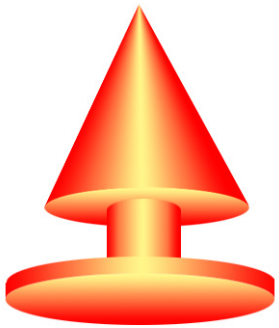
pixelated image

$$I_{DS}(r,c)=\frac{1}{\Delta^2}\sum_{\rho=r\Delta}^{(r+1)\Delta-1}\sum_{\chi=c\Delta}^{(c+1)\Delta-1} I(\rho,\chi); \quad I_P = I_{DS}\uparrow\Delta$$

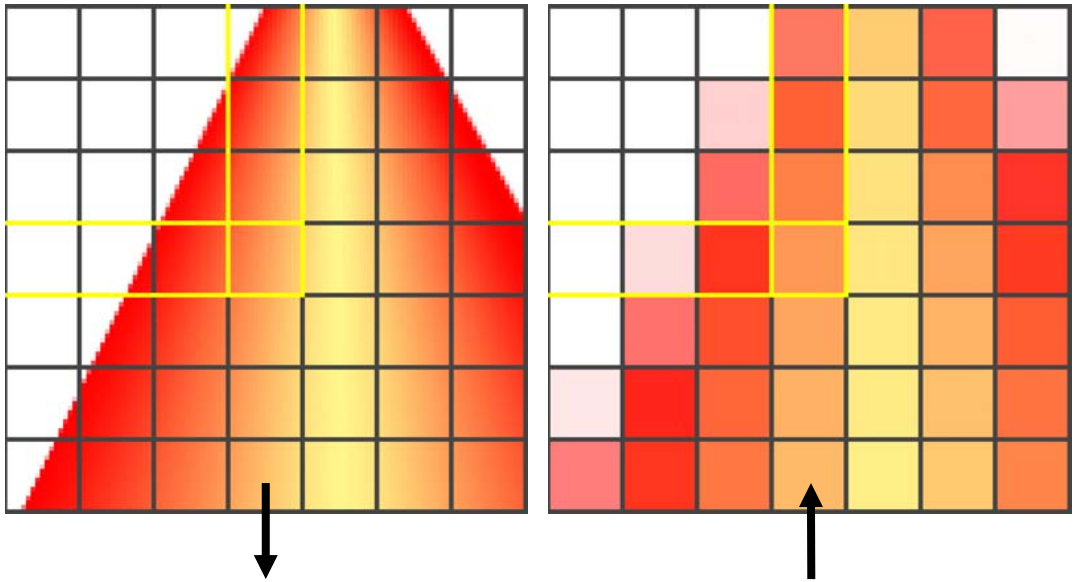
(*upsampled*)

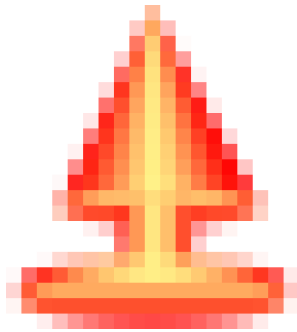
Pixelization

Take the average within each square.



$I(\rho,\chi)$





$I_P(r,c)$

high-res image

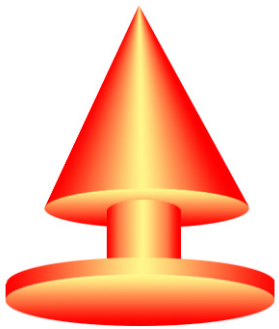
$$I_{DS}(r,c) = \frac{1}{\Delta^2} \sum_{\rho=r\Delta}^{(r+1)\Delta-1} \sum_{\chi=c\Delta}^{(c+1)\Delta-1} I(\rho,\chi); \quad I_P = I_{DS} \uparrow \Delta$$

(*upsampled*)

pixelated image

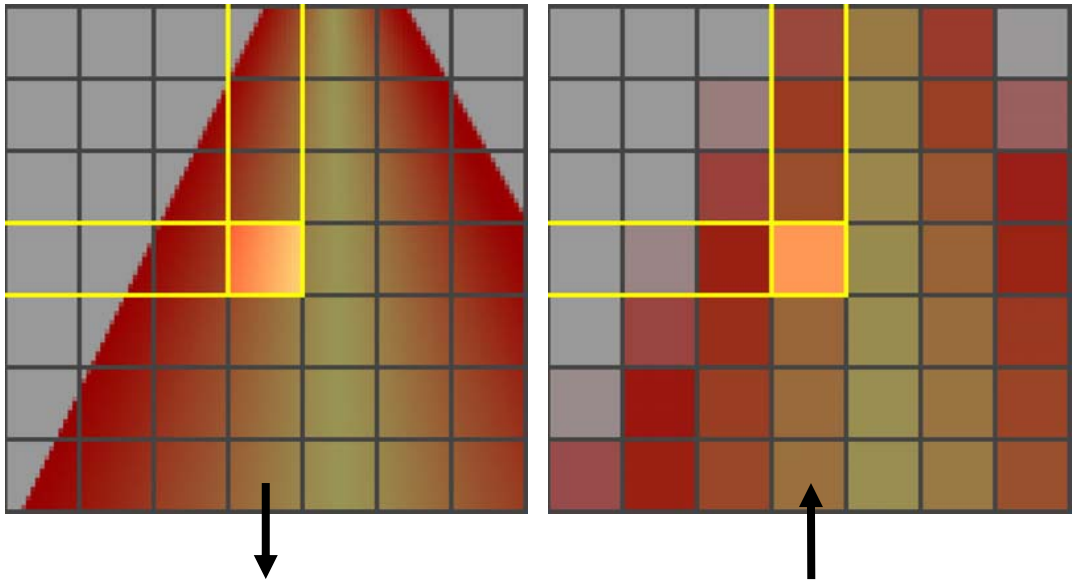
Pixelization

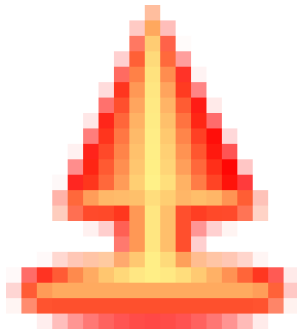
Take the average within each square.



$I(\rho,\chi)$

high-res image





$I_P(r,c)$

pixelated image

$$I_{DS}(r,c) = \frac{1}{\Delta^2} \sum_{\rho=r\Delta}^{(r+1)\Delta-1} \sum_{\chi=c\Delta}^{(c+1)\Delta-1} I(\rho,\chi); \quad I_P = I_{DS} \uparrow \Delta$$

(*upsampled*)

Pixelization

original image

Pixelization

8 of 8: 256x256

Pixelization

7 of 8: 128x128

Pixelization

6 of 8: 64x64

Pixelization

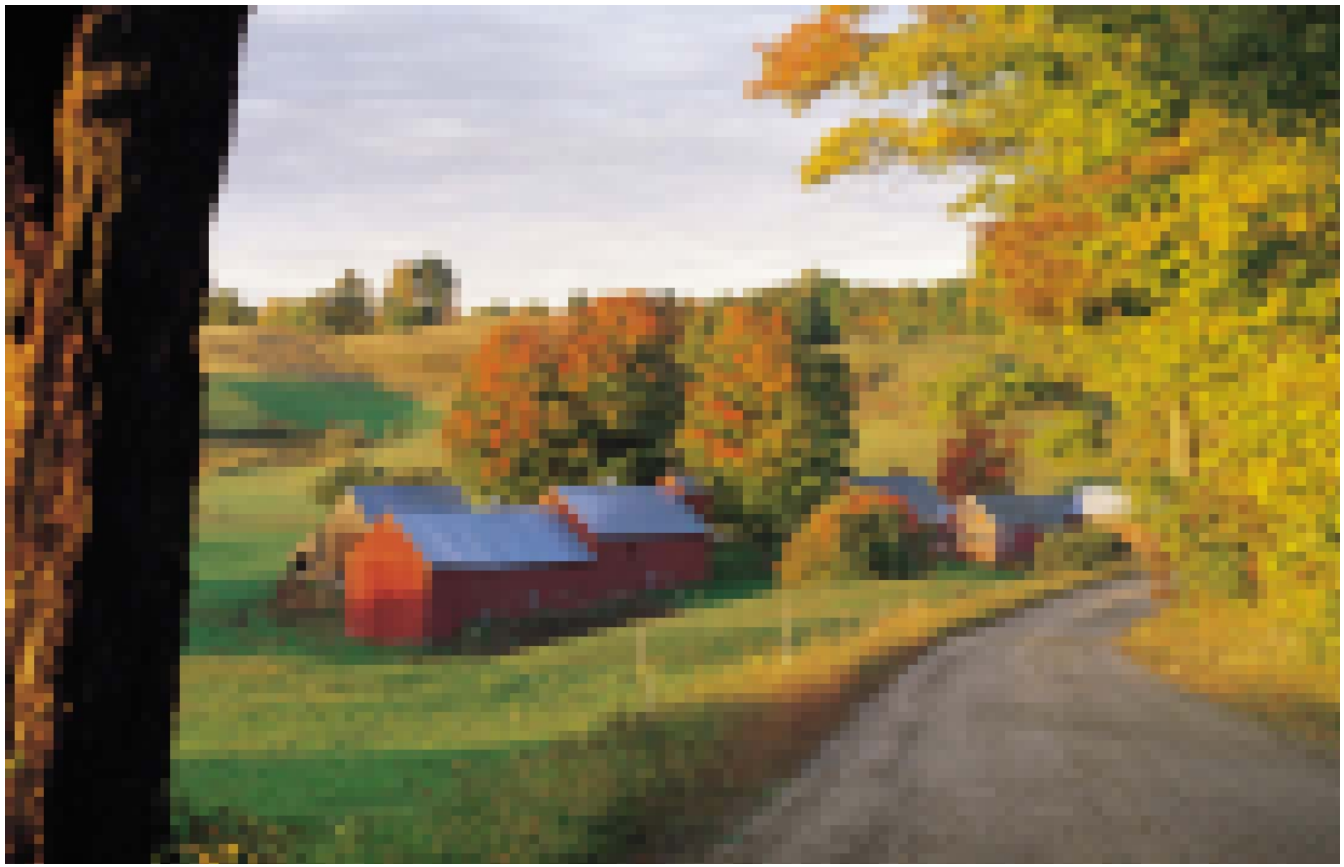
5 of 8: 32x32

Pixelization

4 of 8: 16x16

Pixelization

3 of 8: 8x8



Pixelization

2 of 8: 4x4

Pixelization

1 of 8: 2x2

Pixelization : cross-fade

original image

Cross-fade: a*mask + (1-a)*original

Pixelization by Factor 32

Cross-fade, a=5/5

Cross-fade: a*mask + (1-a)*original

Pixelization by Factor 32

Cross-fade, a=4/5

Cross-fade: a*mask + (1-a)*original

Pixelization by Factor 32

Cross-fade, a=3/5

Cross-fade: a*mask + (1-a)*original

Pixelization by Factor 32

Cross-fade, a=2/5

Cross-fade: a*mask + (1-a)*original

Pixelization by Factor 32

Cross-fade, a=1/5

Cross-fade: a*mask + (1-a)*original

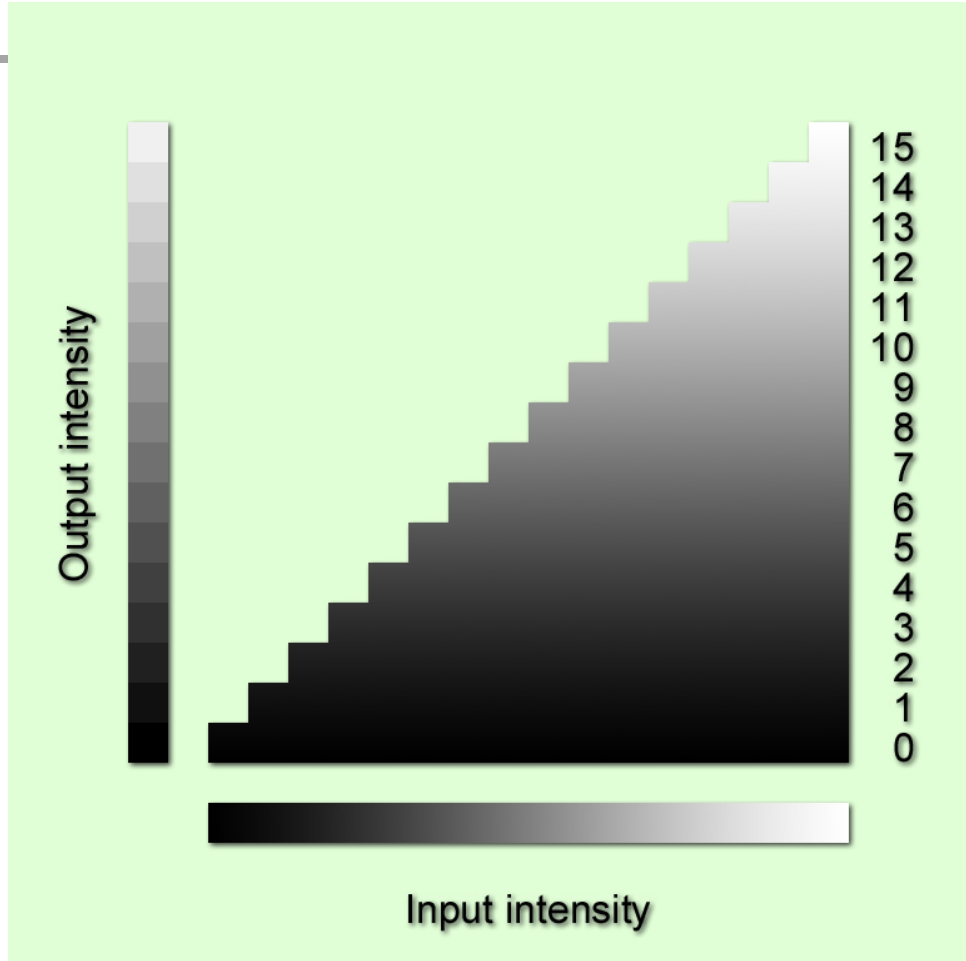
Pixelization by Factor 32

original image

Quantization

16 million colors

16 colors



Quantization

8 bits 256 levels

7 bits 128 levels

6 bits 64 levels

5 bits 32 levels

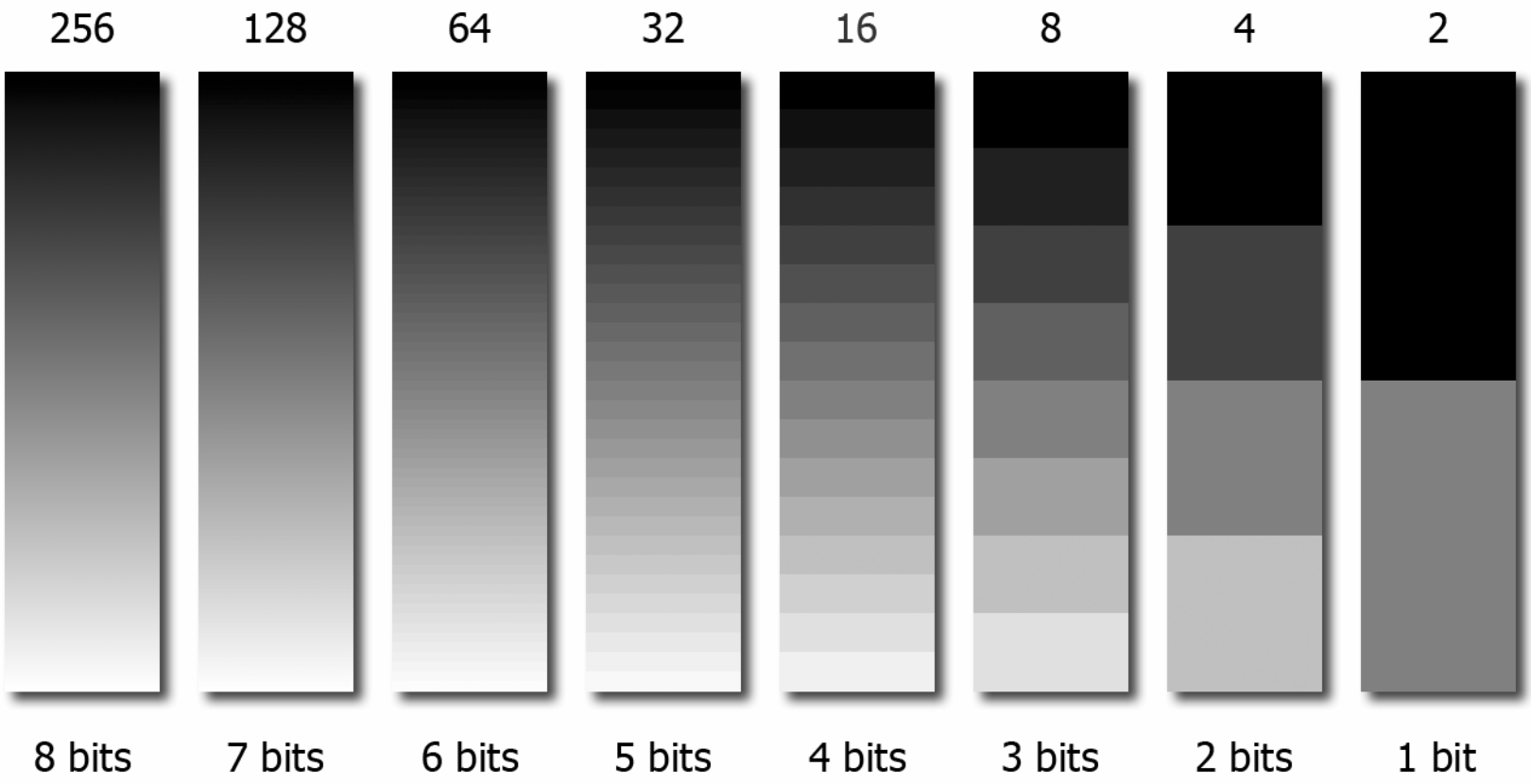
4 bits 16 levels

3 bits 8 levels

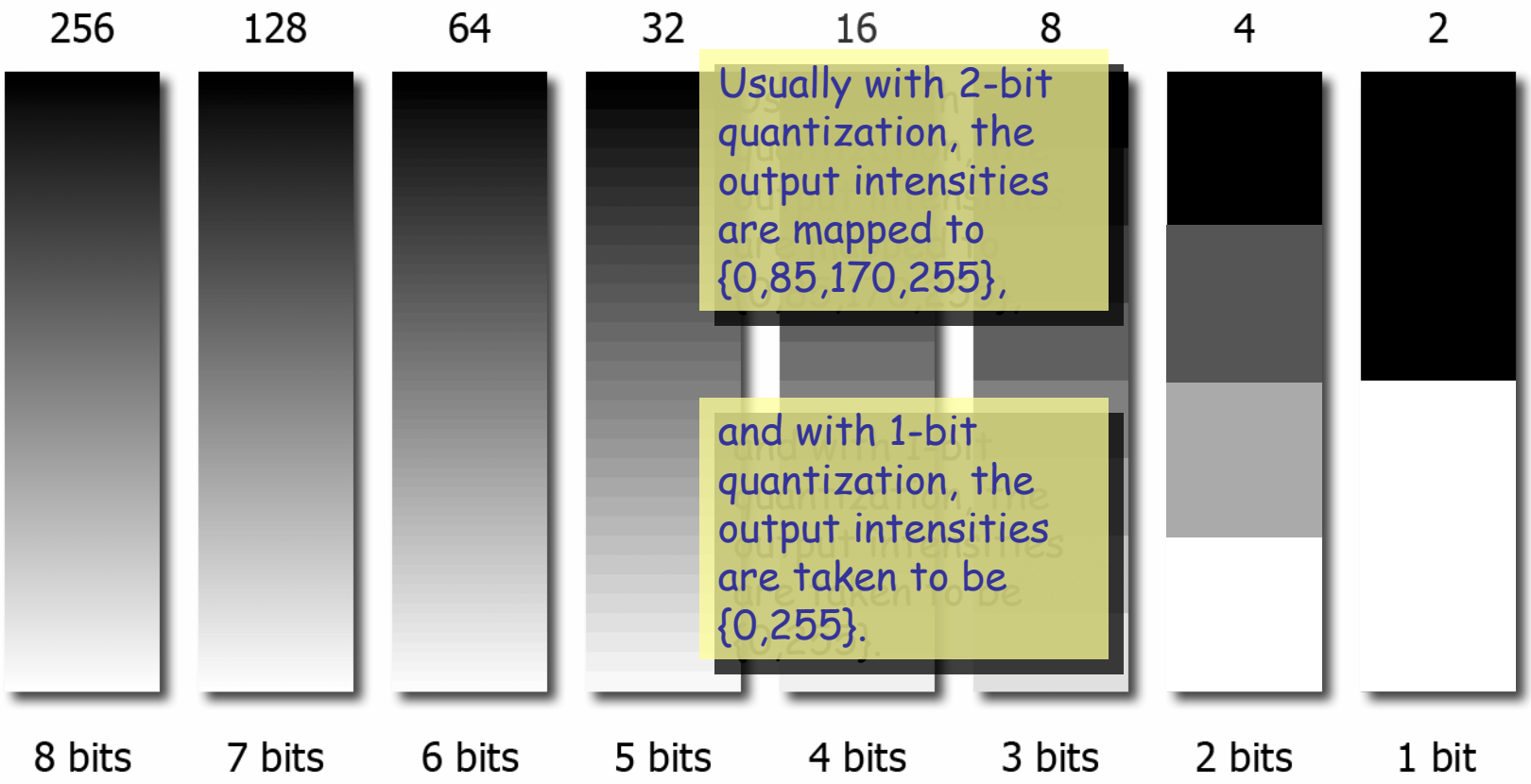
2 bits 4 levels

1 bit 2 levels

Intensity Quantization

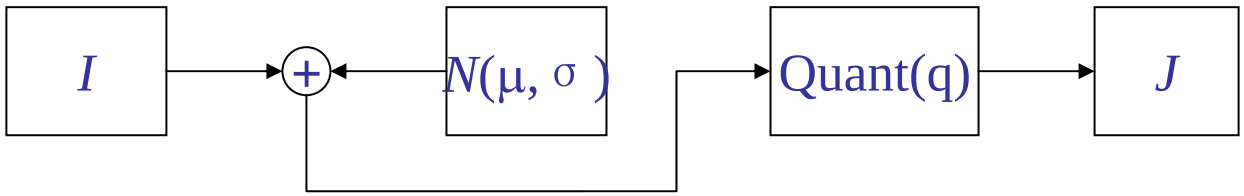


Intensity Quantization

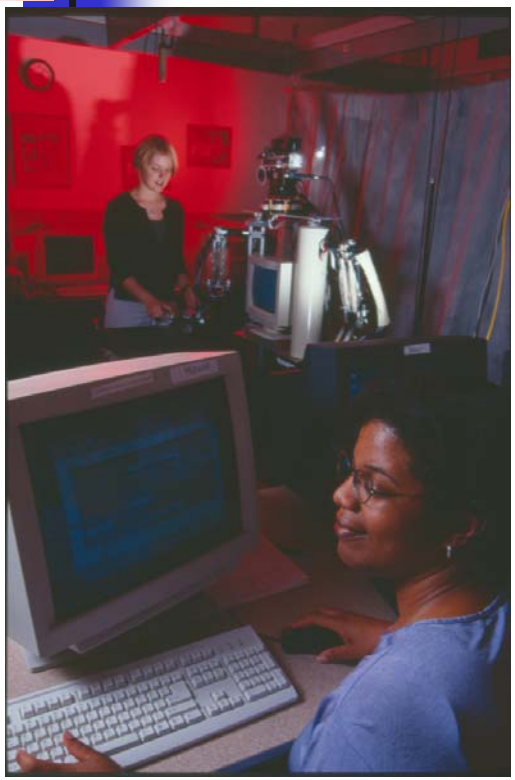


Dithering(抖动): Noise Improves Quantization

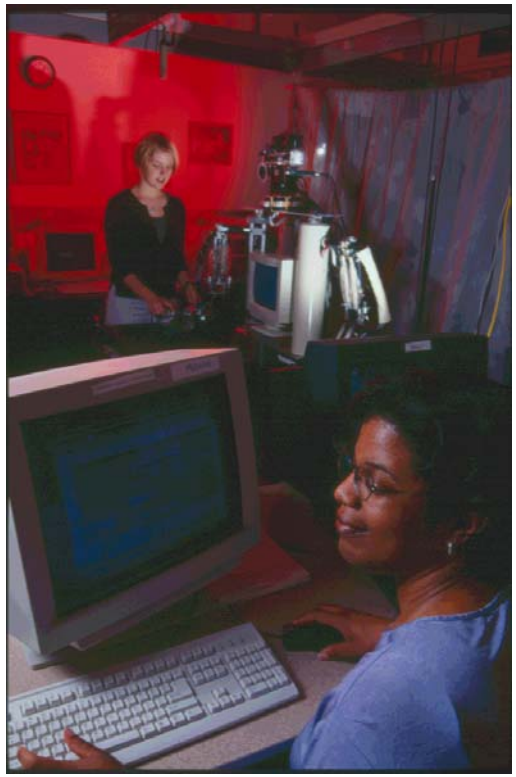
- Quantizing an image into 1, 2, or 3 bits can introduce false contours.
- The addition of signed noise to the image before quantization can improve the appearance of the result. This is called *dithering*.
- The noise usually should have $\mu = 0$.
- The σ of the noise must be determined through experimentation since it depends on the image being quantized. A reasonable first choice for uniformly distributed noise is $\sigma = \frac{1}{4}M/q$, where M is the maximum intensity value in the image (*e.g.* 255) and q is the number of bits in the quantized image.



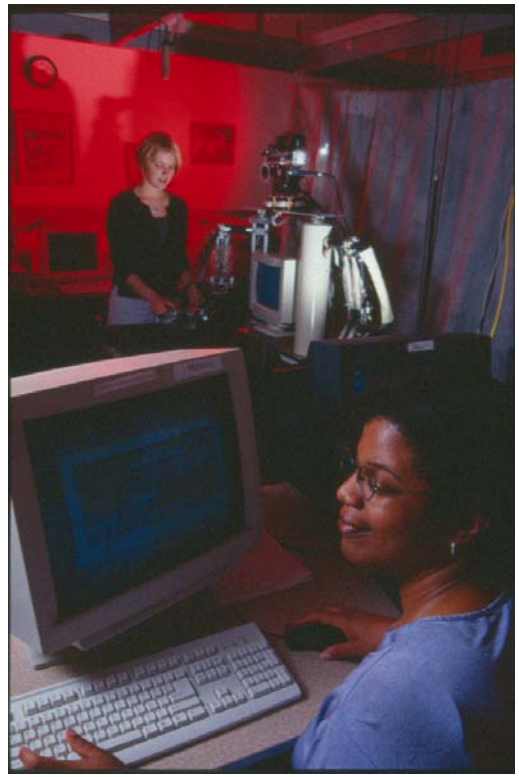
Dithering:
use noise to reduce quant. error



8 bits

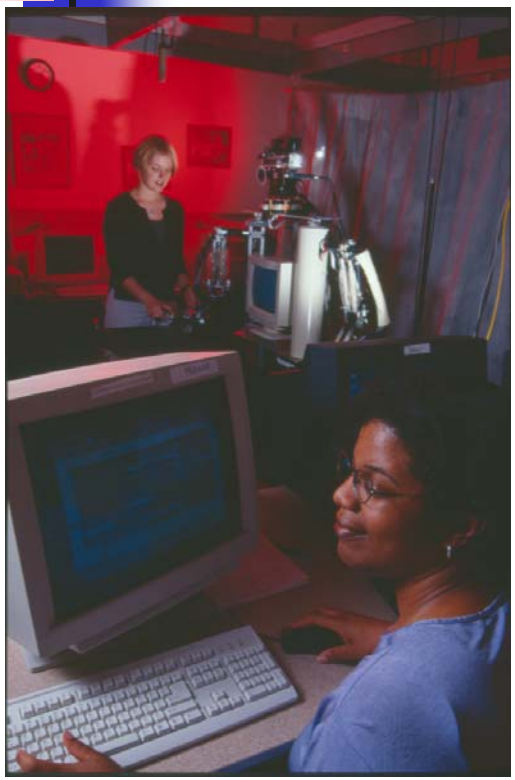


4 bits



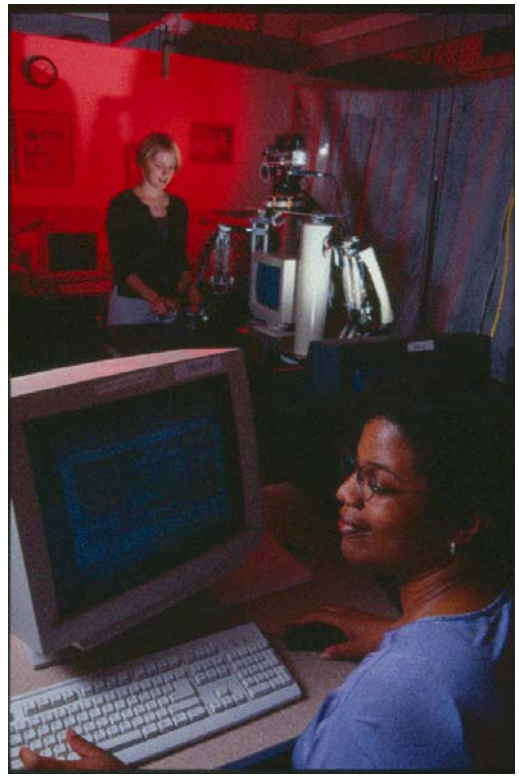
4 bits + noise

Dithering:
use noise to reduce quant. error



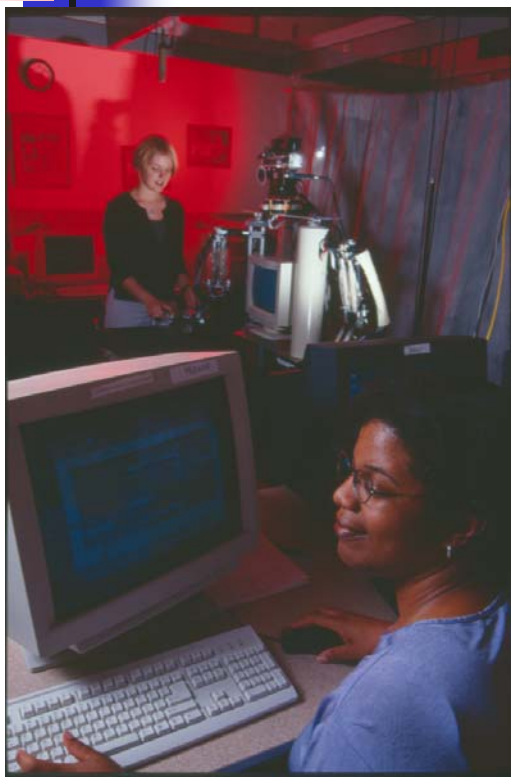
8 bits

3 bits



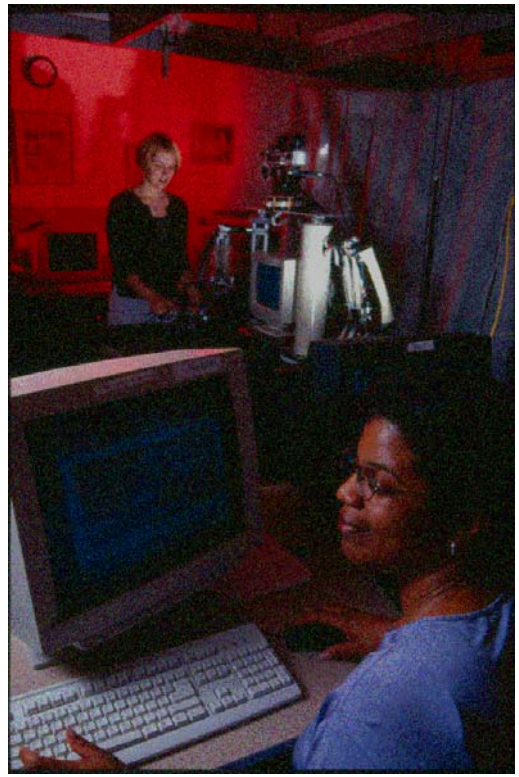
3 bits + noise

Dithering:
use noise to reduce quant. error



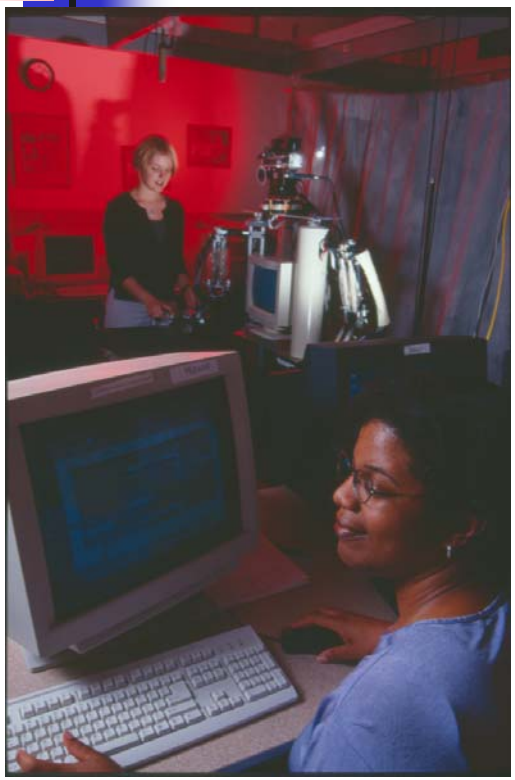
8 bits

2 bits



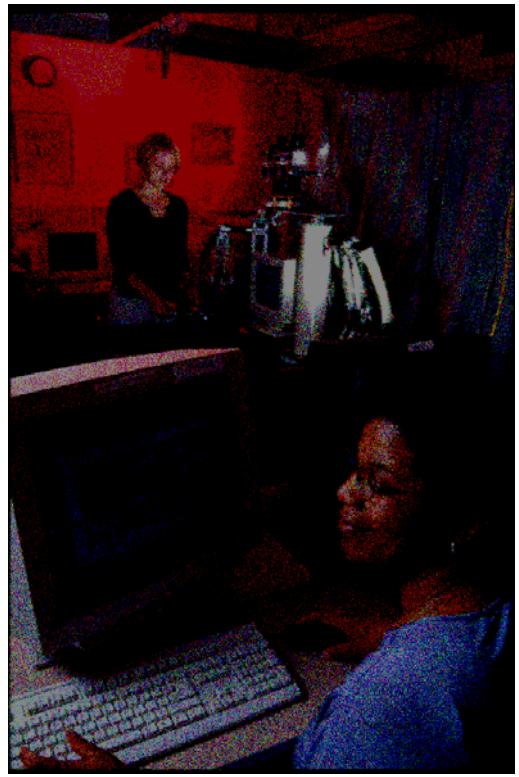
2 bits + noise

Dithering:
use noise to reduce quant. error



8 bits

1 bit



1 bit + noise

Dithering: Matlab demo -- Grayscale image

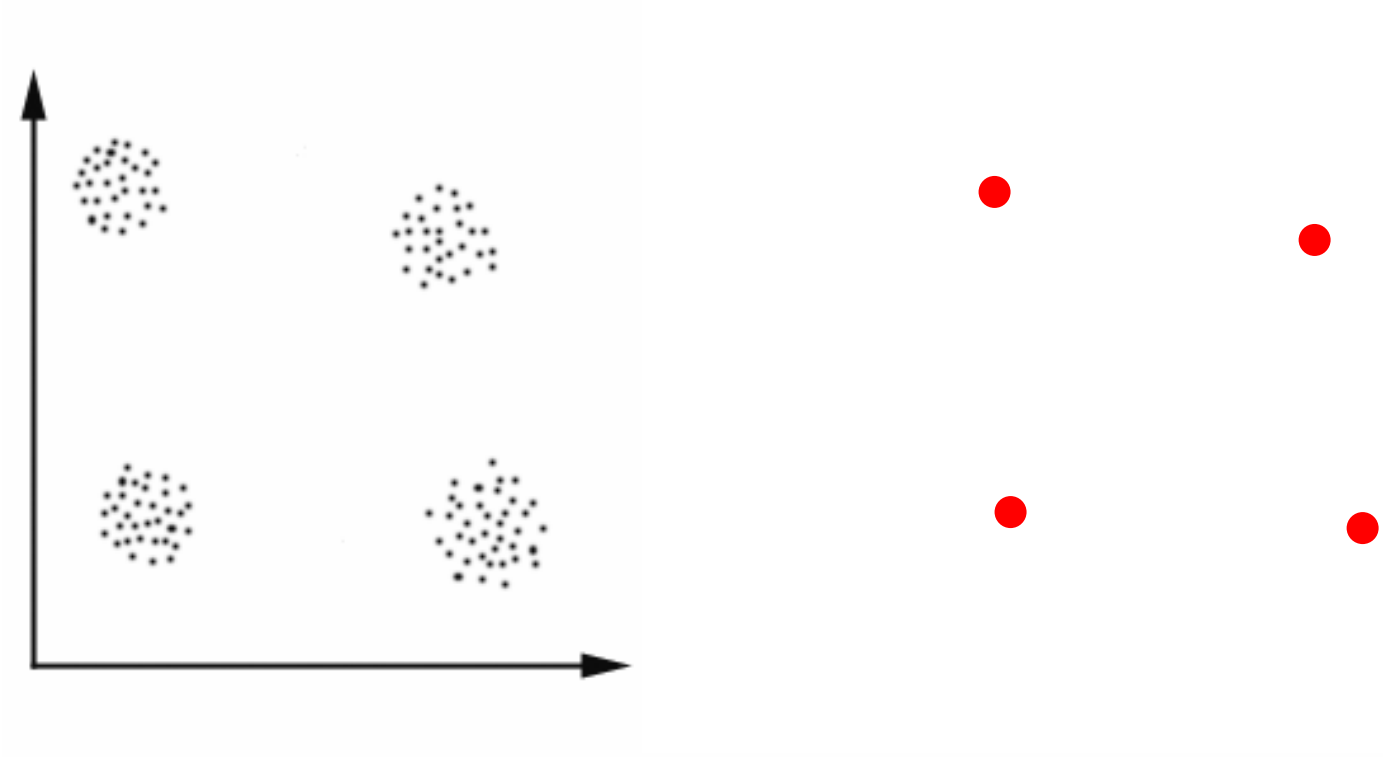
```
clear all; close all;
iptsetpref('ImshowBorder','tight');

I = imread('cameraman.tif');
figure,imshow(I);

BW1 = im2bw(I,0.5);
figure, imshow(BW1);

BW2 = dither(I);
figure, imshow(BW2);
```

Dithering: Matlab demo -- Color image

```
rgb=imread('onion.png');
height=size(rgb,1);width=size(rgb,2);
figure,imshow(rgb);

map=colormap(colorcube(64));
BWrgb = rgb2ind(rgb,map,'nodither');
figure,imshow(BWrgb,map);

noise=30.0*randn(height,width,3);
rgbnoise=uint8(double(rgb)+noise);
BWrgbnoise = rgb2ind(rgbnoise,map,'nodither');
figure,imshow(BWrgbnoise,map);
% The following two lines are equivalent
% DITrgb =rgb2ind(rgb,map,'dither');
DITrgb = dither(rgb,map);
figure,imshow(DITrgb,map);

[X_no_dither,map]= rgb2ind(rgb,64,'nodither');
figure, imshow(X_no_dither,map);

[X_dither,map]=rgb2ind(rgb,64,'dither');
figure, imshow(X_dither,map);
```

How to choose optimal index colors ?

Clustering Principle



Application of Quantization: Steganography(信息隐藏)

8-bit-per-band, 3-band, "original" image

Pieter Bruegel (the Elder, ca. 1525-69), *The Peasant Dance*, 1568, Oil on oak panel, 114x164 cm, Kunsthistorisches Museum Wien, Vienna

If an image is quantized, say from 8 bits to 6 bits and redisplayed it can be all but impossible to tell the difference between the two.

Application of Quantization: Steganography

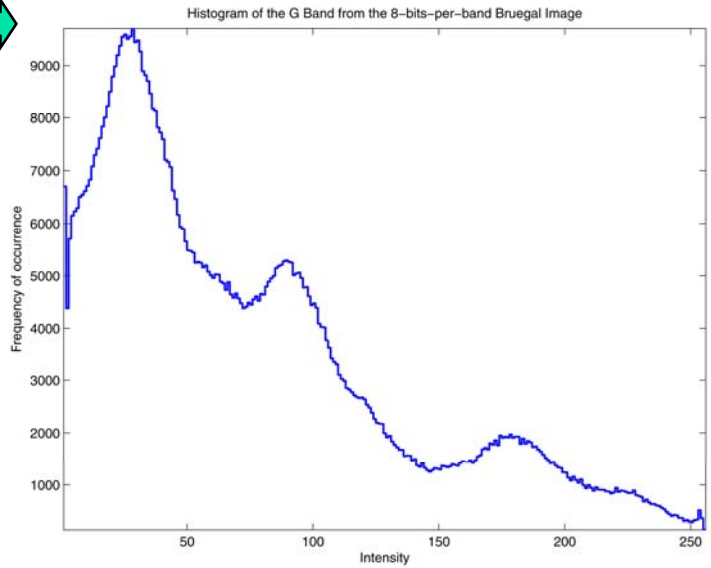
6-bit-per-band, 3-band, quantized image

Pieter Bruegel (the Elder, ca. 1525-69), *The Peasant Dance*, 1568, Oil on oak panel, 114x164 cm, Kunsthistorisches Museum Wien, Vienna

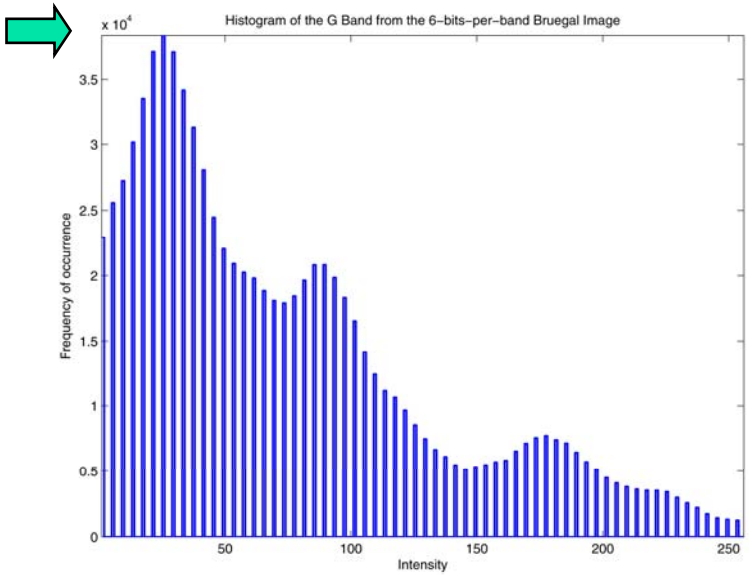
If an image is quantized, say from 8 bits to 6 bits and redisplayed it can be all but impossible to tell the difference between the two.

Application of Quantization: Steganography

green-band histogram of 8-bit image



green-band histogram of 6-bit image

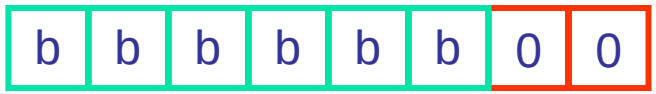


The histograms of the two versions indicate which is which. If the 6-bit version is displayed as an 8-bit image, it has only pixels with values 0, 4, 8, … , 252

0, 4, 8, … , 252

Application of Quantization: Steganography

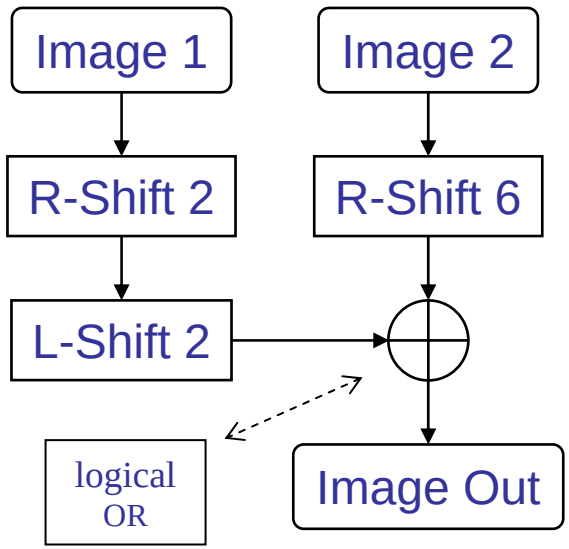
If the 6-bit version is displayed as an 8-bit image then the 8-bit pixels all have zeros in the lower 2 bits:



b = 0 or 1 always 0

This introduces the possibility of encoding other information in the low-order bits.

That other information could be a message, perhaps encrypted, or even another image.



X-Shift *n* = logical left or right shift by *n* bits.

Application of Quantization: Steganography

Image 1 in upper 6-bits.
Image 2 in lower 2-bits.

Pieter Bruegel (the Elder, ca. 1525-69), *The Peasant Dance*, 1568, Oil on oak panel, 114x164 cm, Kunsthistorisches Museum Wien, Vienna

The second image is invisible because the value of each pixel is between 0 and 3. For any given pixel, its value is added to the to the collocated pixel in the first image that has a value from the set {0, 4, 8, … , 252}. The 2nd image is noise on the 1st.

Application of Quantization: Steganography

Image 1 in upper 6-bits.
Image 2 in lower 2-bits.

Pieter Bruegel (the Elder, ca. 1525-69), *The Peasant Dance*, 1568, Oil on oak panel, 114x164 cm, Kunsthistorisches Museum Wien, Vienna

L-Shift 6

?

To recover the second image (which is 2 bits per pixel per band) simply left shift the combined image by 6 bits.

Application of Quantization: Steganography

Image 2 in upper 2-bits.
Image 1 shifted out

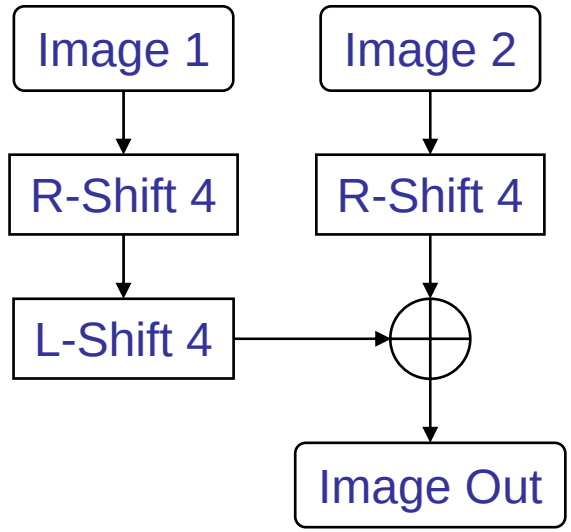
From the video game, *ZeroWing*, by Toaplan.
See http://en.wikipedia.org/wiki/All_your_base

To recover the second image (which is 2 bits per pixel per band) simply left shift the combined image by 6 bits.

Application of Quantization: Steganography

This is so effective that two 4-bit-per-pixel images can be superimposed with only the image in the high-order bits visible. Both images contain the same amount of information but because one takes on values between 0 and 15, the other takes on values from {16, 32, 48, … , 240}, and the smaller values are added to the larger, the image in the low-order bits is effectively invisible

Images 1 and 2 each have 4-bits per pixel when combined.



Application of Quantization: Steganography

Original Image

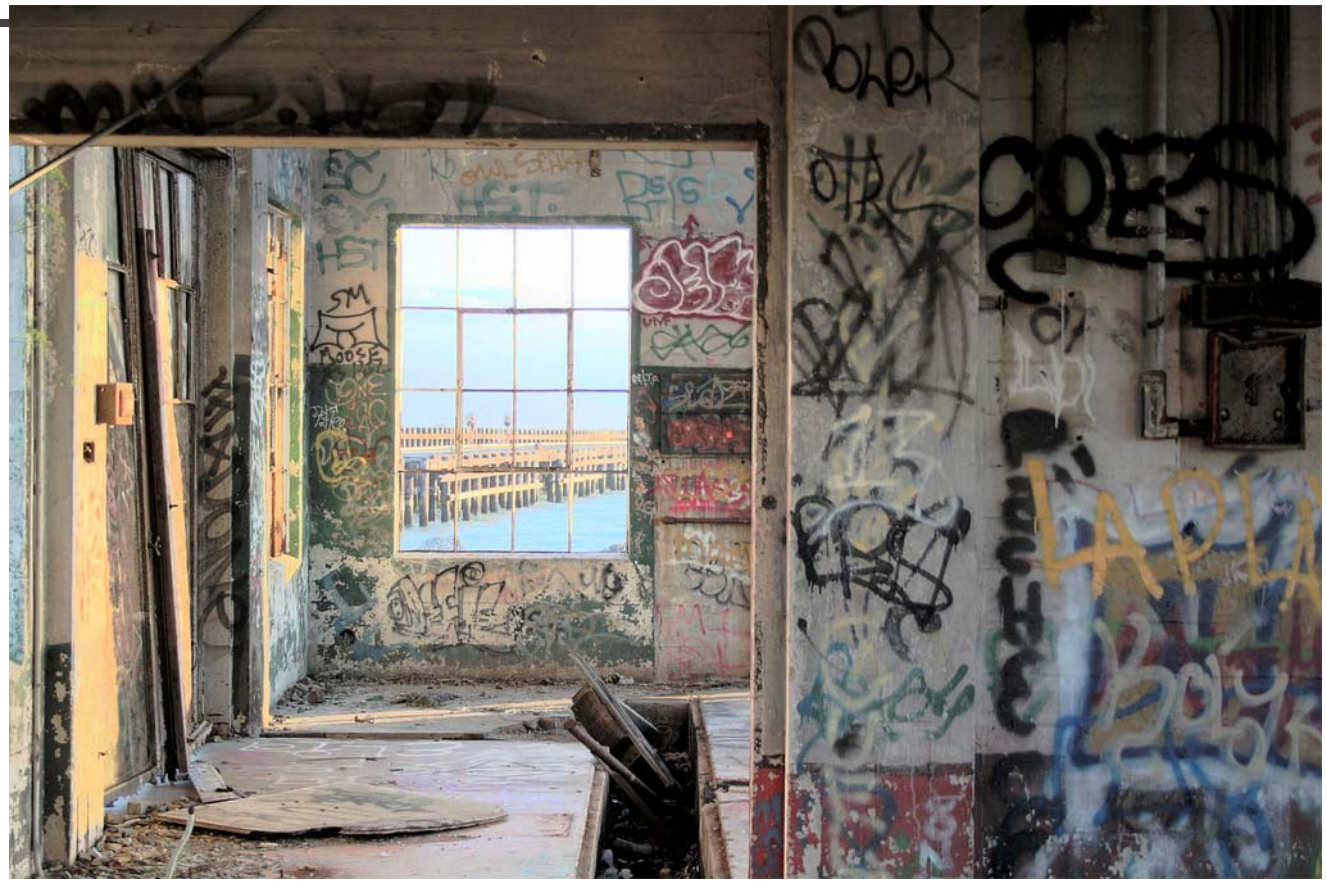


Photo: CypherOne
http://www.flickr.com/people/cypherone/

Application of Quantization: Steganography

Image quantized to 4-bits per pixel.

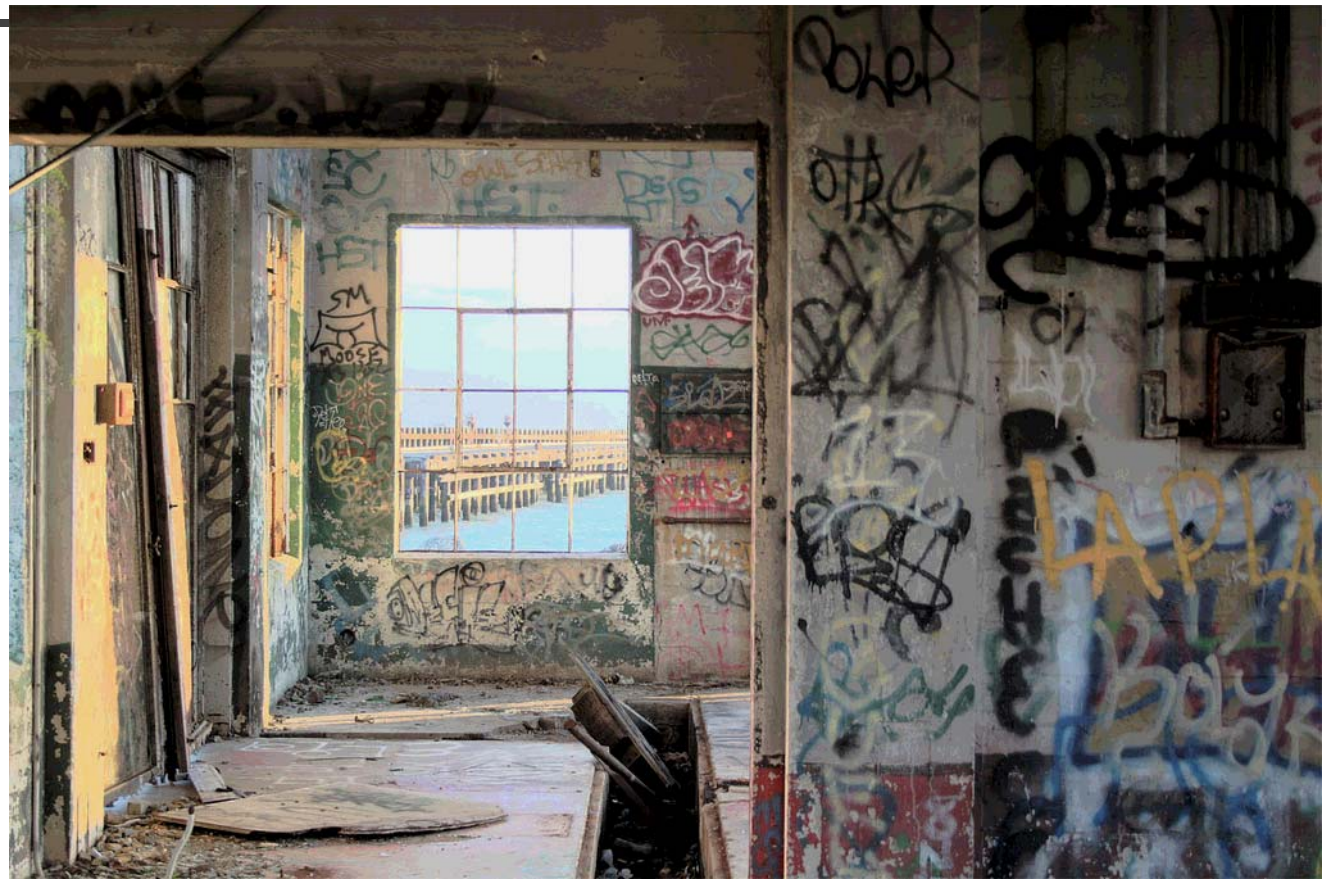


Photo: CypherOne
http://www.flickr.com/people/cypherone/

Application of Quantization: Steganography

Image 1 in upper 4-bits.
Image 2 in lower 4-bits.

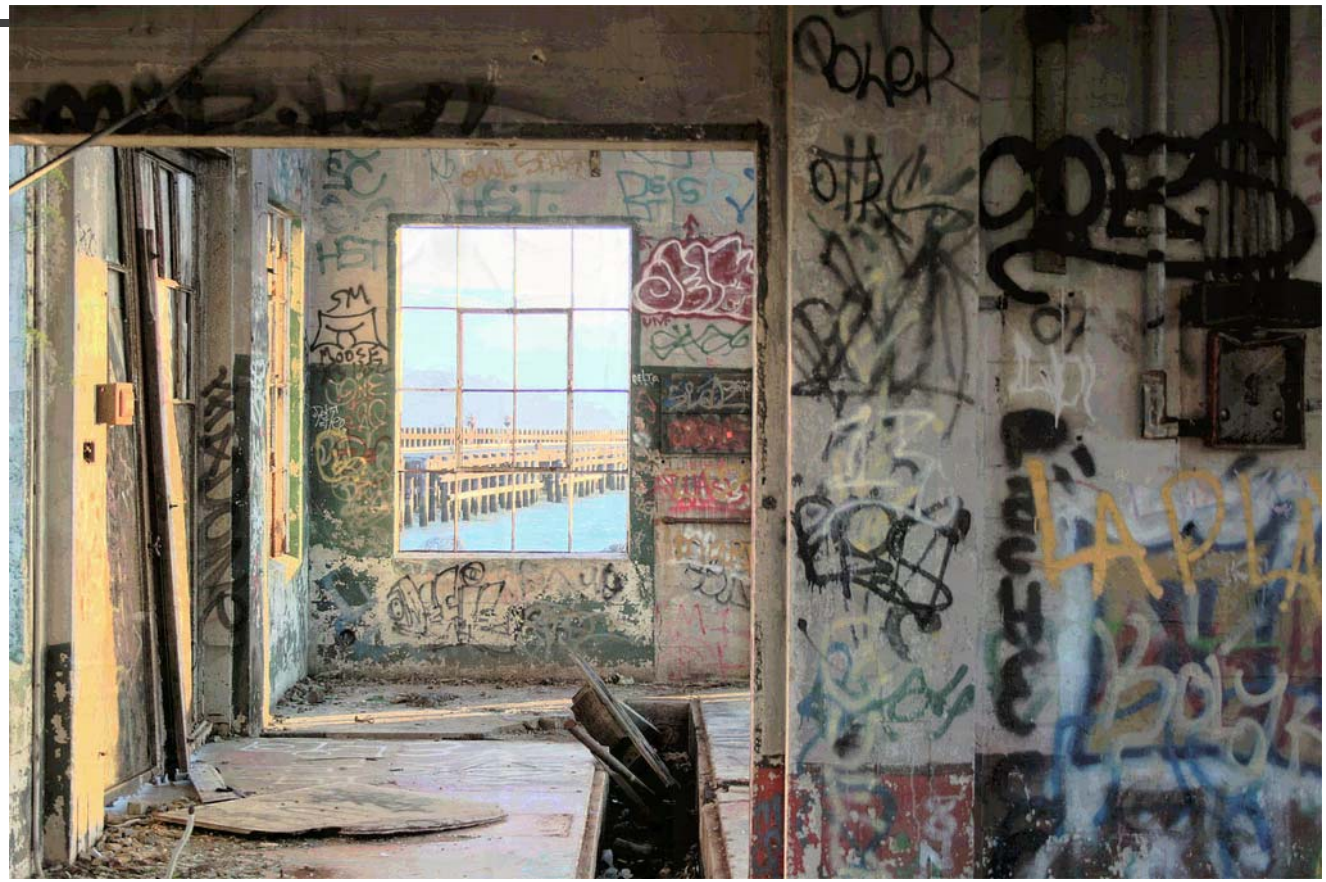


Photo: CypherOne
http://www.flickr.com/people/cypherone/

Application of Quantization: Steganography

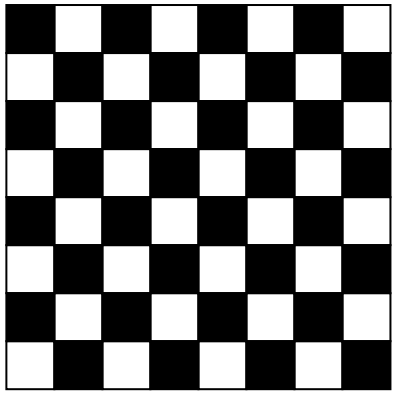
Image 2 in upper 4-bits.
Image 1 shifted out.

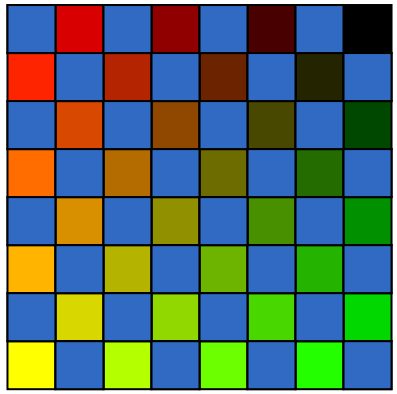
Photographer Unknown

Hidden image creation

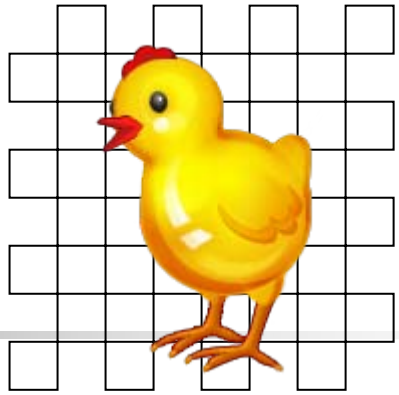
Hidden image creation

- What happens when you press CTRL+a?
  - Assume the image as a chessboard with each pixel act as one squares, black(even-pixel) or white(odd-pixel).
  - All even pixels corresponding to black squares are masked with the color (49,106,197).





Hidden image creation



- Basic idea
  - Put background image into odd pixels
  - Put foreground image into even pixels
  - Foreground image should be strong enough to mask background image
  - Background image should be strong enough to show up when foreground image is masked
    - *What kinds of image to choose?*
    - *Brightness adjustment*
    - *Contrast adjustment*
    - *Saturation adjustment*
    - *……*

Hidden image creation

- An example

Challenge – From grayscale to color

Texture transfer

- We want to add color to the right grayscale image to make it have the style of the left color image

Texture transfer

- Work in Lab space instead of RGB space.
- Use local brightness to search for best match.
- Copy the color components from the source image to the target image.

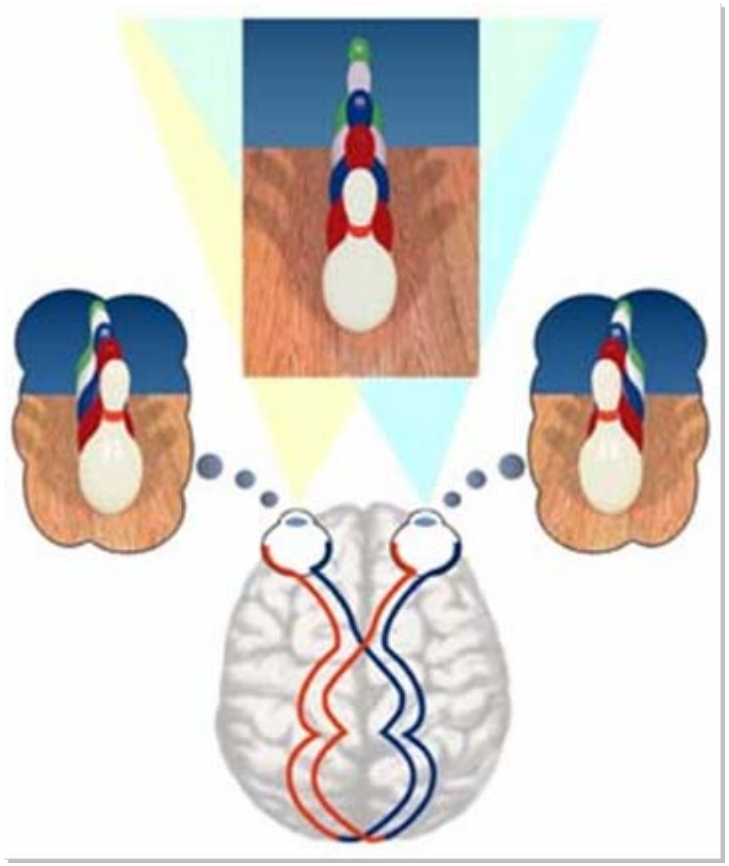
三维立体画——Stereogram

http://www.magiceye.com/

三维立体画——Stereogram

- 三维立体图的英文名称为Stereogram，根据图种类的不同还有其他的名称，
  - RDS（Random Dot Stereogram）
  - AutoStereogram /SIRDS（Single Image Random
  - Dot Stereogram）、
  - SIRTS（Single Image Random TEXT Stereogram，或asciistereograms)、
  - Hollusion、
  - SIS（Single Image Stereograms）等。
- 三维立体图是一种不需要任何设备、仪器就可以从中看到虚拟的三维立体图象的二维图片。

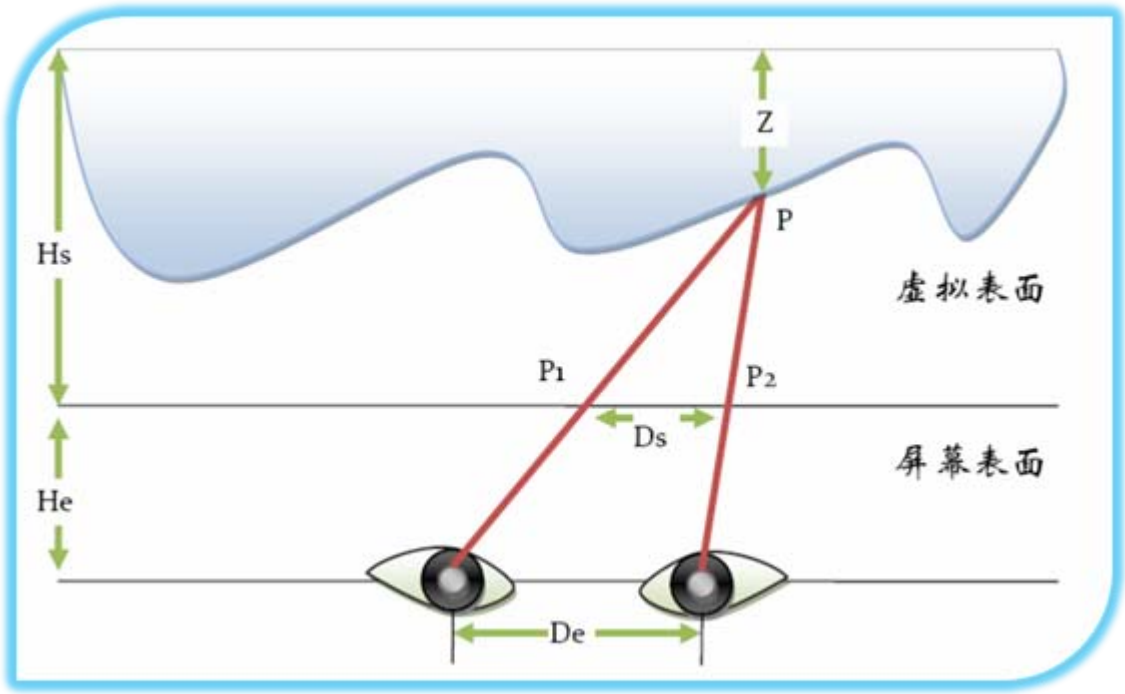
http://www.vision3D.com/stereo.html



三维立体画——Stereogram

$$\frac{Ds}{De} = \frac{Hs - Z}{Hs + He - Z}$$

$$Ds = \frac{Hs - Z}{Hs + He - Z} \times De$$



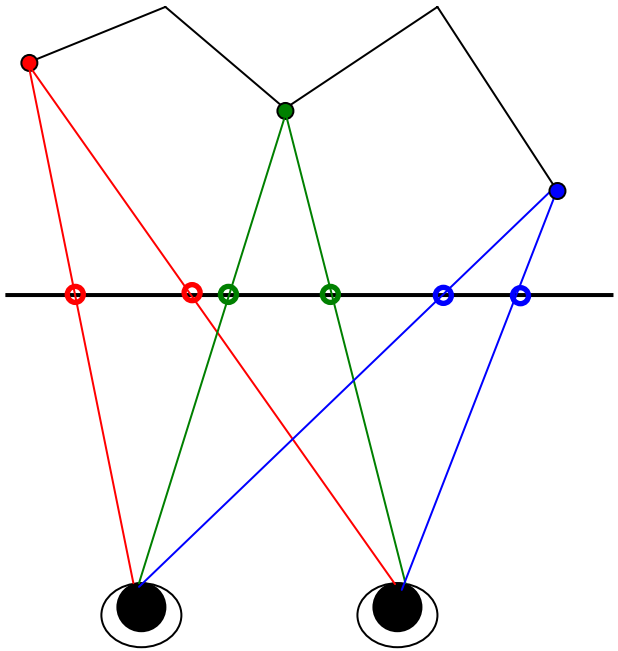
Given fixed *Hs*, *He* and *De*, *Ds* is only dependent on *Z*

制作立体画原理

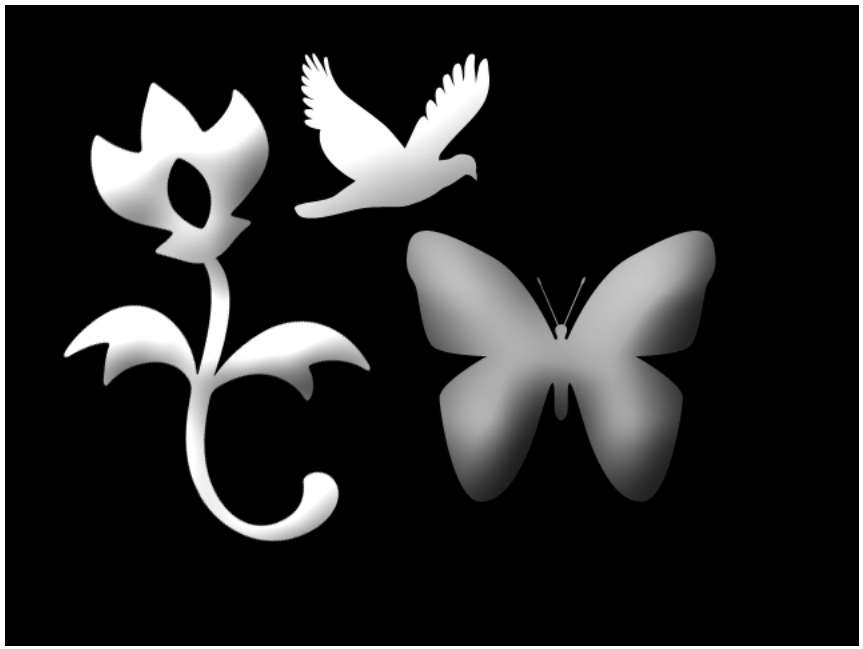
- 表达层次感
  - 在该程序中为了表达物体的空间层次感，我利用图像的灰度值的大小来表达相同点之间的差距，因此灰度值不同的点产生的△X就不相同，这样根据视差得到的空间距离就不同，这样就能够使物体有比较好的层次感。

三维立体画——Stereogram

- 视差



三维立体画——Stereogram



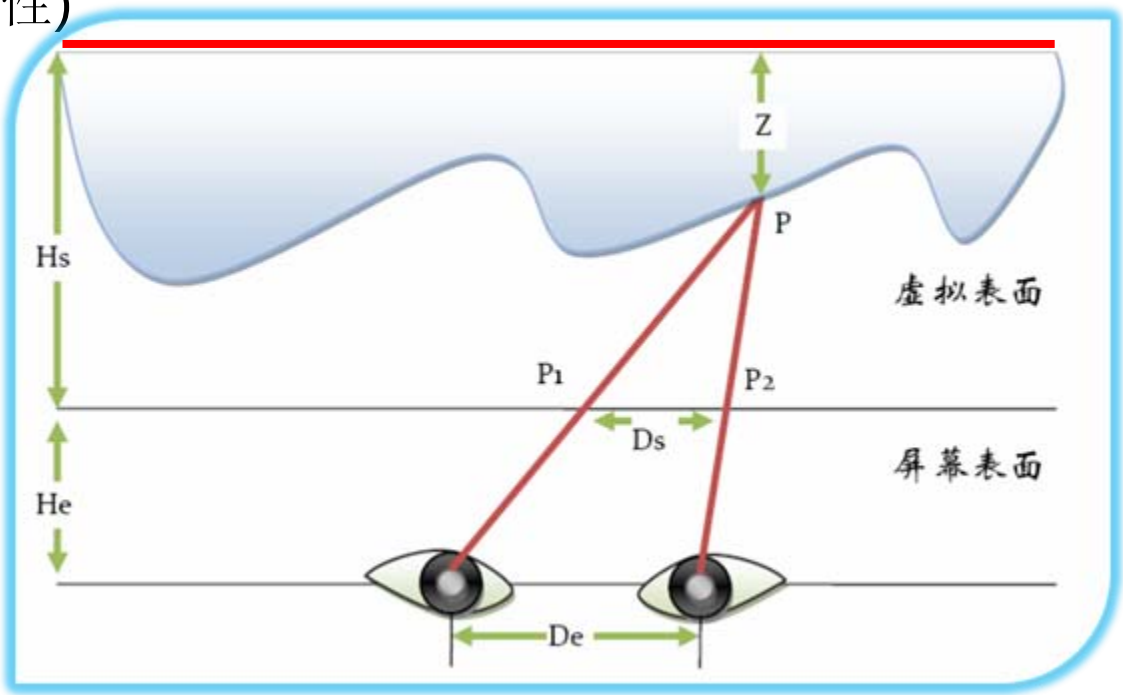
```
BYTE *bytePixels = new BYTE[pH * pWB]; //暂存数组
bmp->GetBitmapBits(pH*pWB, bytePixels);
srand(time(0));
for(i = 0; i < pH * pWB; i += 3)
{
   bytePixels[i] = rand() % 255;
   bytePixels[i + 1] = rand() % 255;
   bytePixels[i + 2] = rand() % 255;
}
   //将像素颜色数组设置到位图资源中
SetBitmapBits(m_hBmpShow, rcHeight * rcWidth * 3, bytePixels);
```

```
color = m_cDepthBmp.GetPixel(i*nBgBmpWidth+x,y);
grey = int(float(GetRValue(color))*0.59+
           float(GetGValue(color))*0.3+
           float(GetBValue(color))*0.11+0.5f);
grey = (LONG)(grey / Depth3D);
if (x+grey>=nBgBmpWidth)
{
   newcolor = cBGBmp.GetPixel(x + grey - nBgBmpWidth, y);
   cBGBmp.SetPixel(x, y, newcolor);
}
else
{
   newcolor = cBGBmp.GetPixel(x + grey, y);
   cBGBmp.SetPixel(x, y, newcolor);
}
```

三维立体画——Stereogram

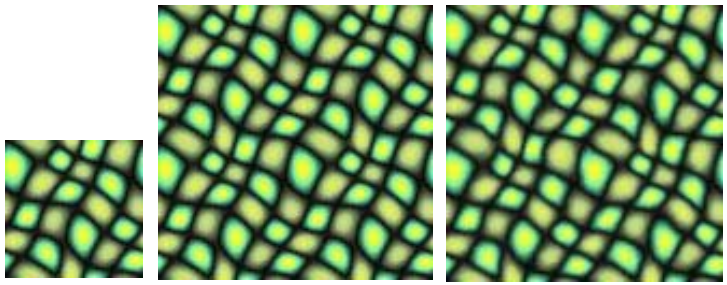
- The property of the background
  - Periodic(周期性)
  - Random(随机性)
  - Tileable(可拼接性)

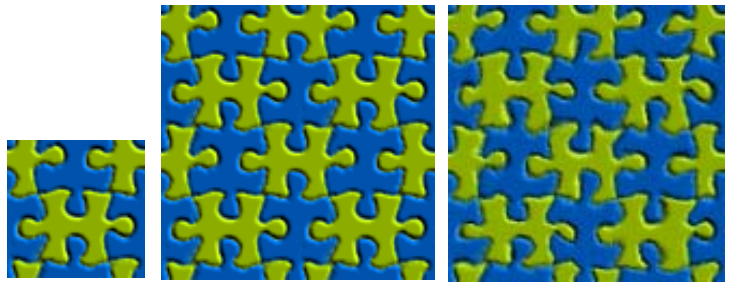
$$\frac{Ds}{De} = \frac{Hs}{Hs + He}$$



Textures

2D texture synthesis





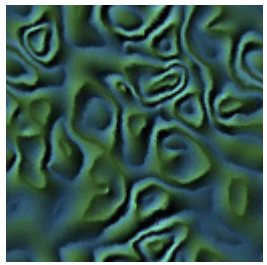


Image Inpainting

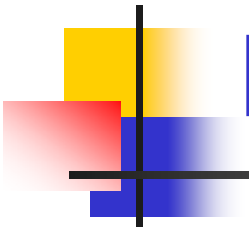
Photo repair

Image Inpainting Example

scene edit

Image Inpainting

Original scene recovery



Homework

- Perform an image editing task. The work can be , but not limited to the following topics:
  - Dithering
  - Segmentation
  - Steganography / Hidden image
  - Texture transfer
  - Stereogram
  - inpainting
  - Others…