Unit 2. Digital Image Representation and Processing

CS 3570
Shang-Hong Lai
CS Dept., NTHU

Content

- Bitmapped images
- Aliasing & Anti-aliasing
- Color models & Color quantization
- Dithering
- Image transform
- Resampling
  - interpolation
- Image Compression
  - Hoffman encoding , JPEG

Bitmaps - digitization

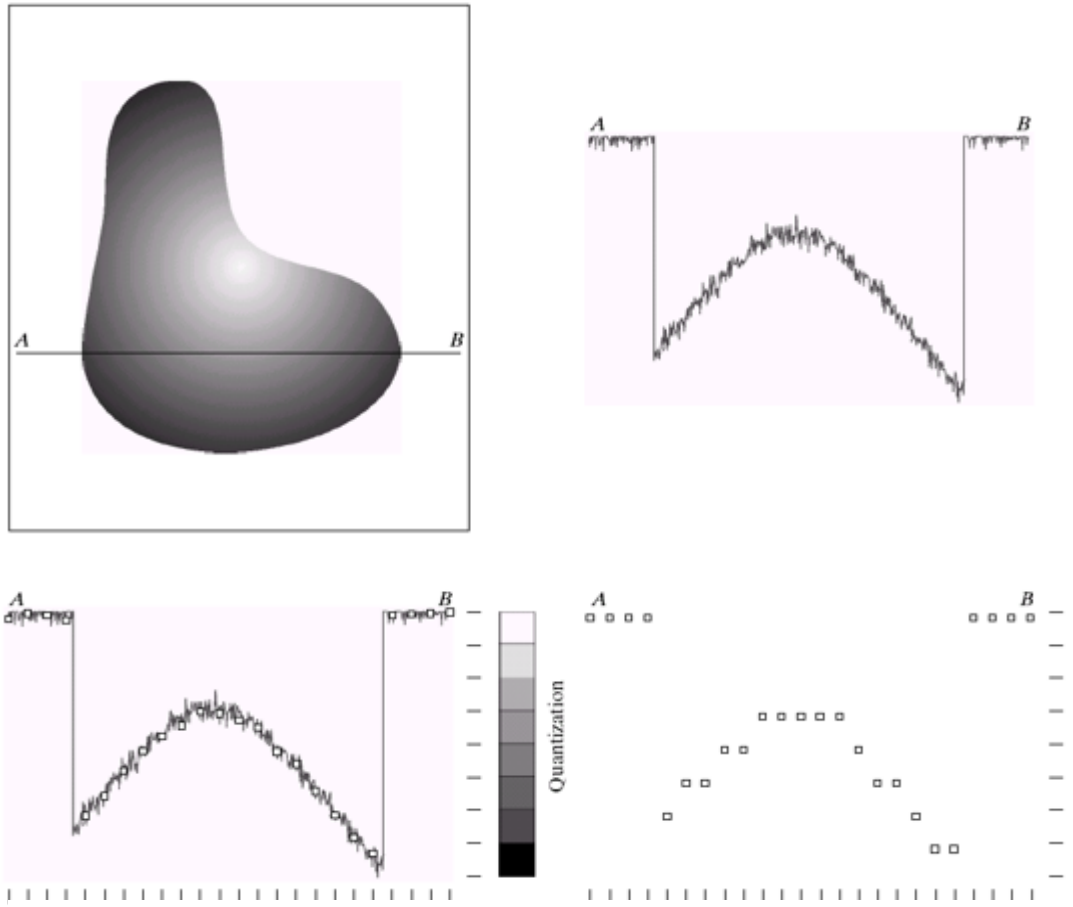
- ***Pixel (picture element)***
  - A number representing the color at position (*r*, *c*) in the bitmap, where *r* is the row and *c* is the column.
- **Sampling**
  - Under-sampled images may contain blocky and unclear effect.
- **Quantization**
  - Low-bit depth can result in patchiness of color

Original

Sampling

Quantization

Bitmaps – Sampling and Quantization



a b
c d

FIGURE 2.16 Generating a digital image. (a) Continuous image. (b) A scan line from *A* to *B* in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

Sampling :

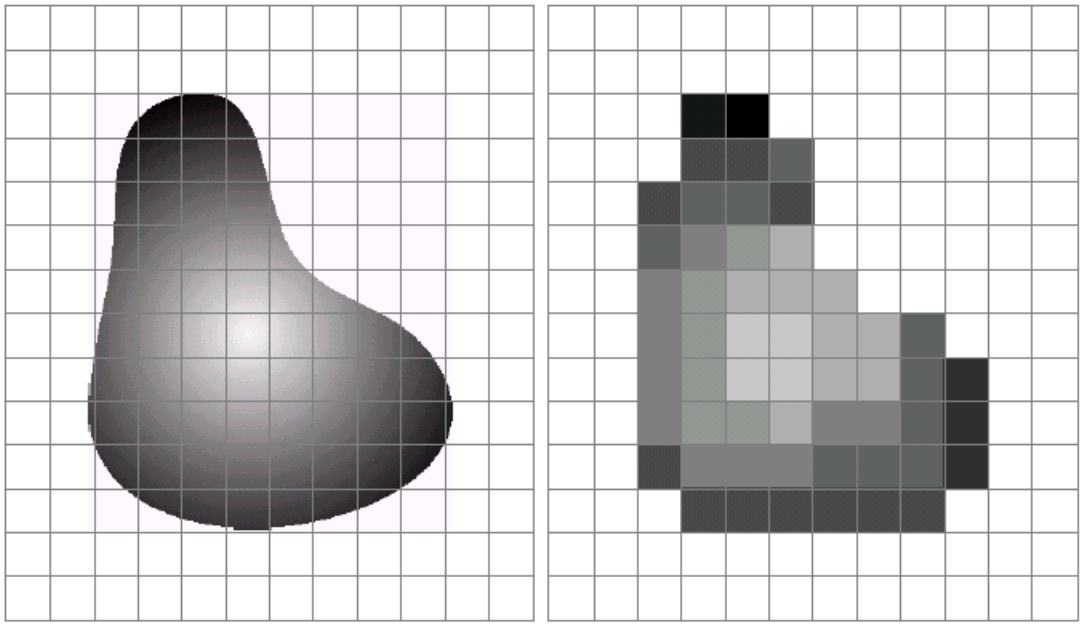
- Digitizing the coordinate values.
- During the sampling step, need to set a sampling rate.

Quantization :

- Digitizing the amplitude values.
- During the quantization step, you need to set bit depth.

- Sampling rate : How frequent to take a data.
- Bit depth: Each sample is represented with a fixed number of bits.

Bitmaps – Sampling and Quantization

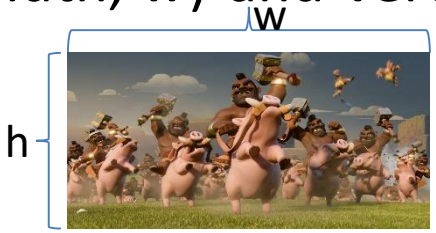


a b

FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

Bitmaps – pixel dimensions & resolution

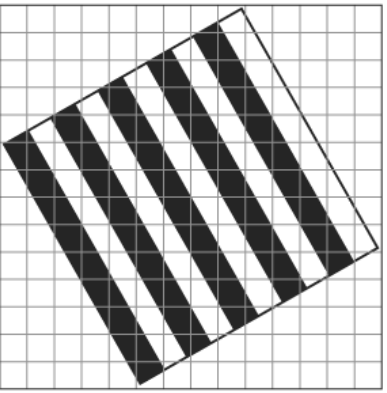
- ***Pixel dimensions***
  - The number of pixels horizontally (*i.e.*, width, *w*) and vertically (*i.e.*, height, *h*), denoted $w \times h$

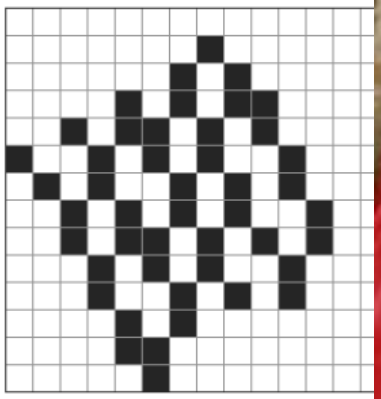


- ***Resolution***
  - The number of pixels in an image per unit of spatial measure.
  - Resolution can be measured in pixels per inch, abbreviated *ppi*. A 200 ppi image will print out using 200 pixels to determine the colors for each inch.
  - Resolution of a printer is a matter of how many dots of color it can print over an area. A common measurement is ***dots per inch*** (DPI).

Aliasing - Moiré patterns

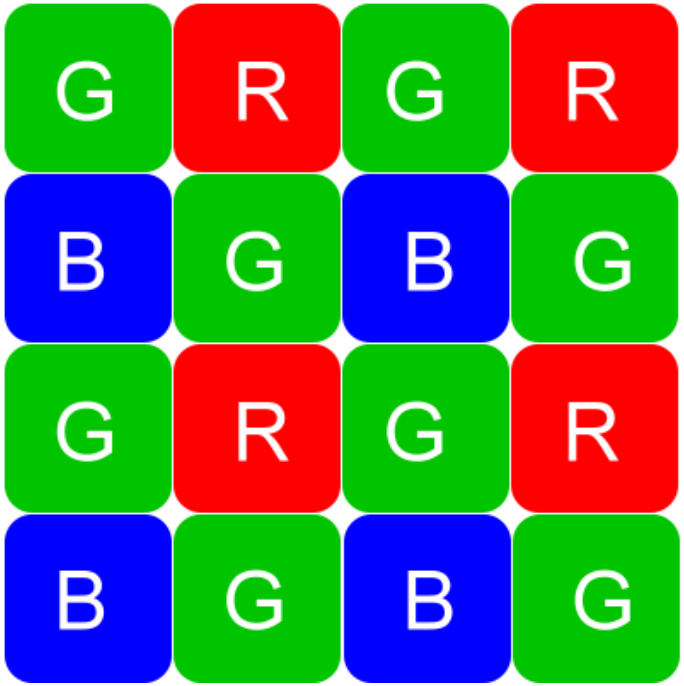
- Aliasing is cause by the discrete nature of pixels *(Sampling Error)*.
- Moiré patterns (Also called moiré effect)
  - The sampling rate for the digital image is not high enough to capture the frequency of the pattern





Bayer Color Filter

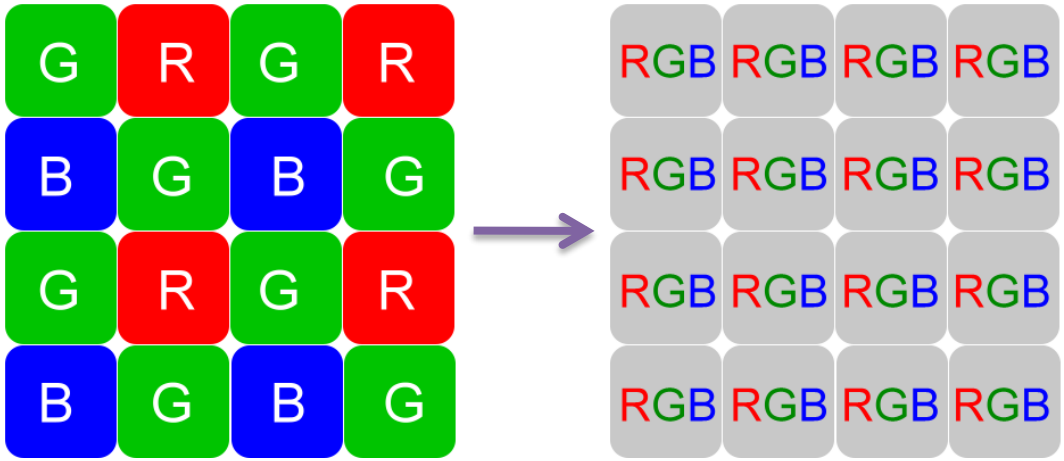
- A less expensive method of color detection uses an array to detect only one color component at each photosite.
- Interpolation is then used to derive the other two color components based on information from neighboring sites.
- A color filter array (CFA) for arranging RGB color filters on a square grid of photosensors.
- Twice as many green sensors as blue or red sensors. Because the human eye is more sensitive to green.



Bayer color filter

Demosaicing

- The interpolation algorithm for deriving the two missing color channels at each photo-site is called ***demosaicing***
- Algorithm
  - Nearest neighbor
  - Linear
  - Cubic
  - Cubic spline

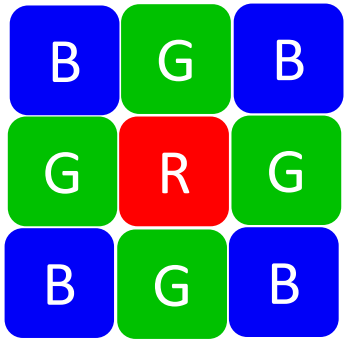


Nearest neighbor algorithm

ALGORITHM 2.1 - NEAREST NEIGHBOR ALGORITHM

```
algorithm nearest_neighbor{
   for each photosite (i,j) where the photosite detects color c1 {
      for each c2 ∈ {red,green,blue} such that c2 ≠ c1 {
         S = the set of nearest neighbors of site (i,j) that have color c2
         set the color value for c2 at site (i,j) equal to the average of the color values
         of c2 at the sites in S
      }
   }
}
```

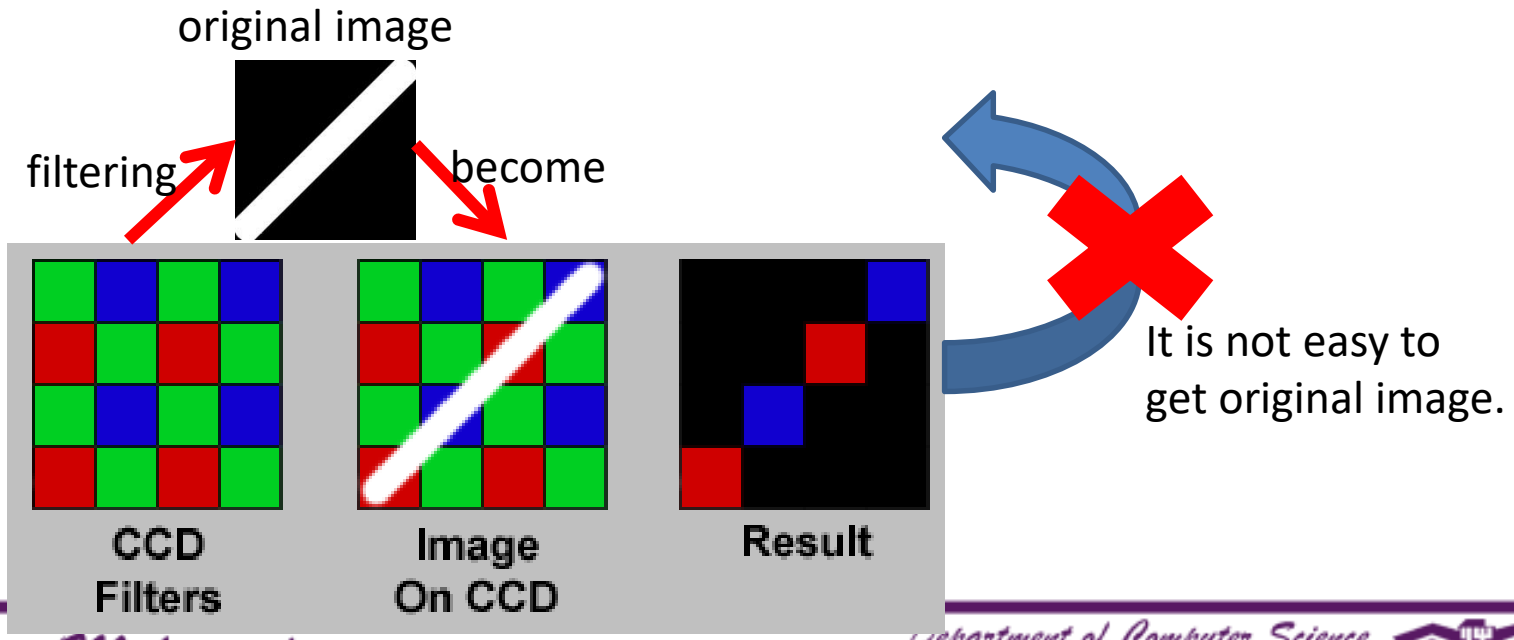
(a) Determining R or B from the center G photosite entails an average of two neighboring sites



(b) Determining B from the center R photosite entails an average of four neighboring sites diagonal from R

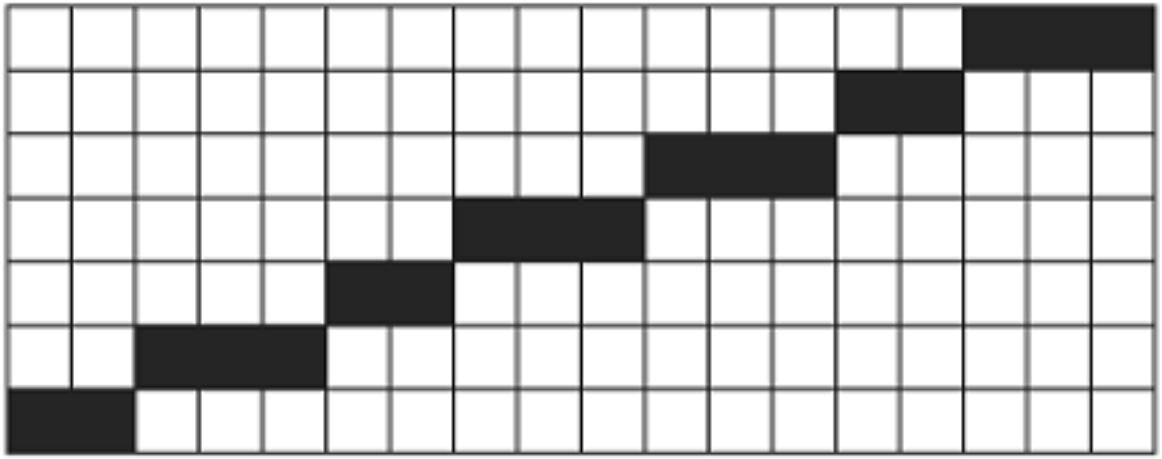
Color Aliasing

- Interpolation can't give a perfect reproduction of the scene being photographed, and occasionally color aliasing results from the process, detected as moiré patterns, streaks, or spots of color not present in the original scene.
- Example: you photograph a white line on the black background



Aliasing - Jagged Edges

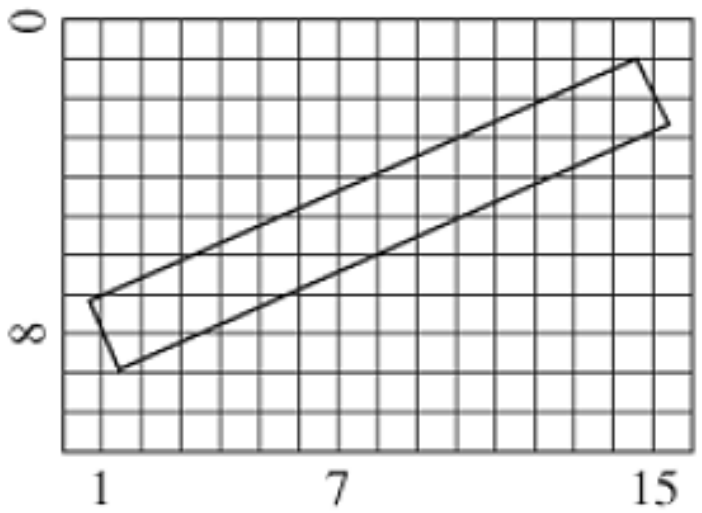
- ***Aliasing*** used to describe the jagged edges along lines or edges that are drawn at an angle across a computer screen
- A line on a computer screen is made up of discrete units: ***pixels***



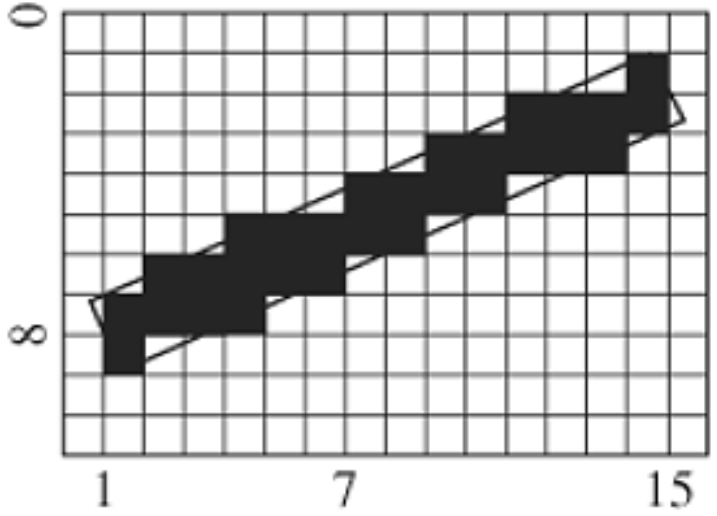
Aliasing of a line that is one pixel wide

Algorithm for drawing a line

- Left figure shows a line that is two pixels wide going from point (8, 1) to point (2, 15)
- In right figure a pixel is colored black if at least half its area is covered by the two-pixel line



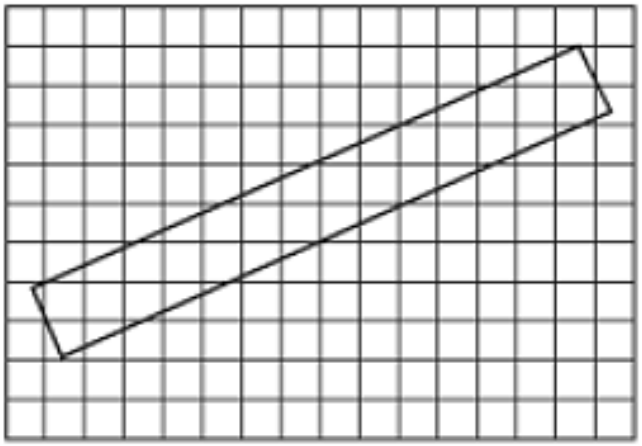
Drawing a line two pixels wide



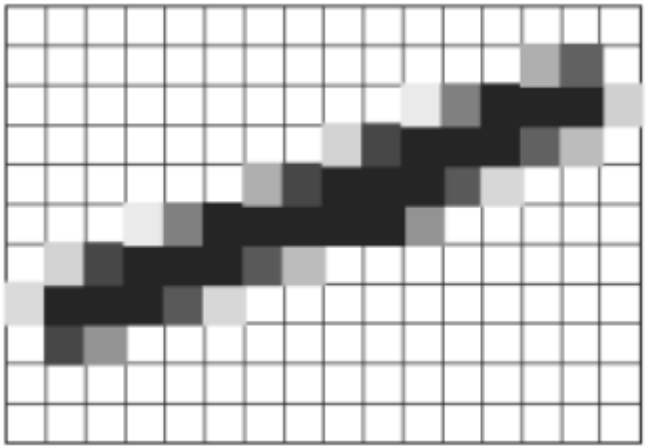
Line two pixels wide, aliased

Anti-aliasing

- **Anti-aliasing** is a technique for reducing the jaggedness of lines or edges caused by aliasing
- The idea is to color a pixel with a shade of its designated color in proportion to the amount of the pixel that is covered by the line or edge



a line two pixels wide



Line two pixels wide, anti-aliased

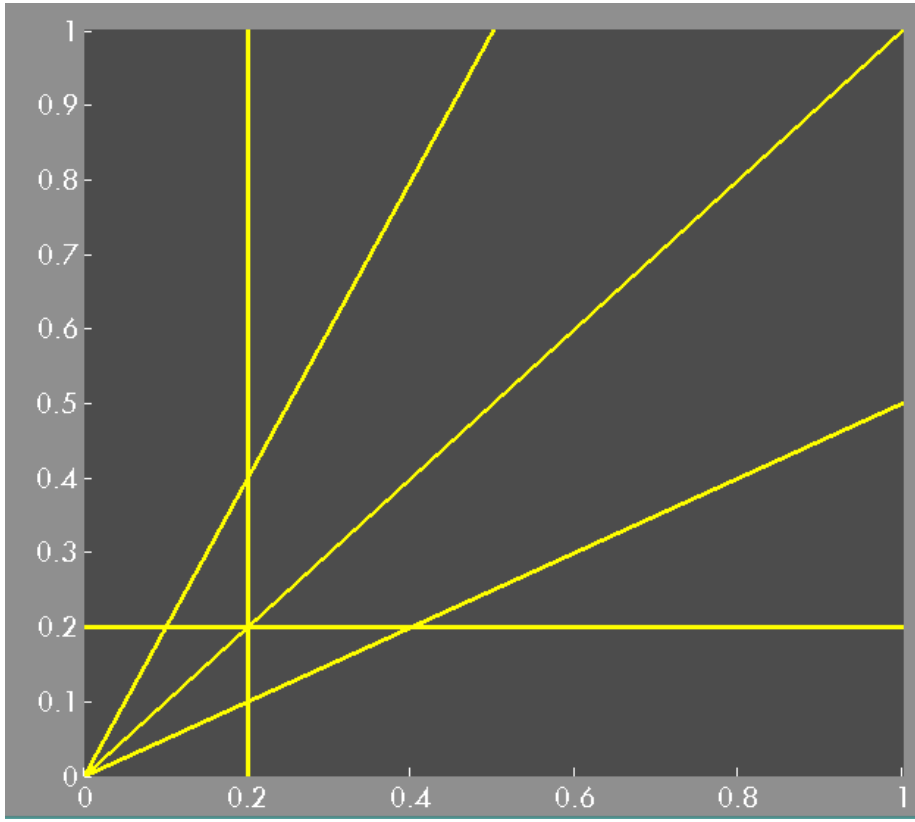
Aliasing of bitmap & vector graph

- Vector graphics suffer less from aliasing problems than do bitmap images in that vector graphics images can be resized without loss of resolution
- **Upsampling**
  - The image is later enlarged by increasing the number of pixels

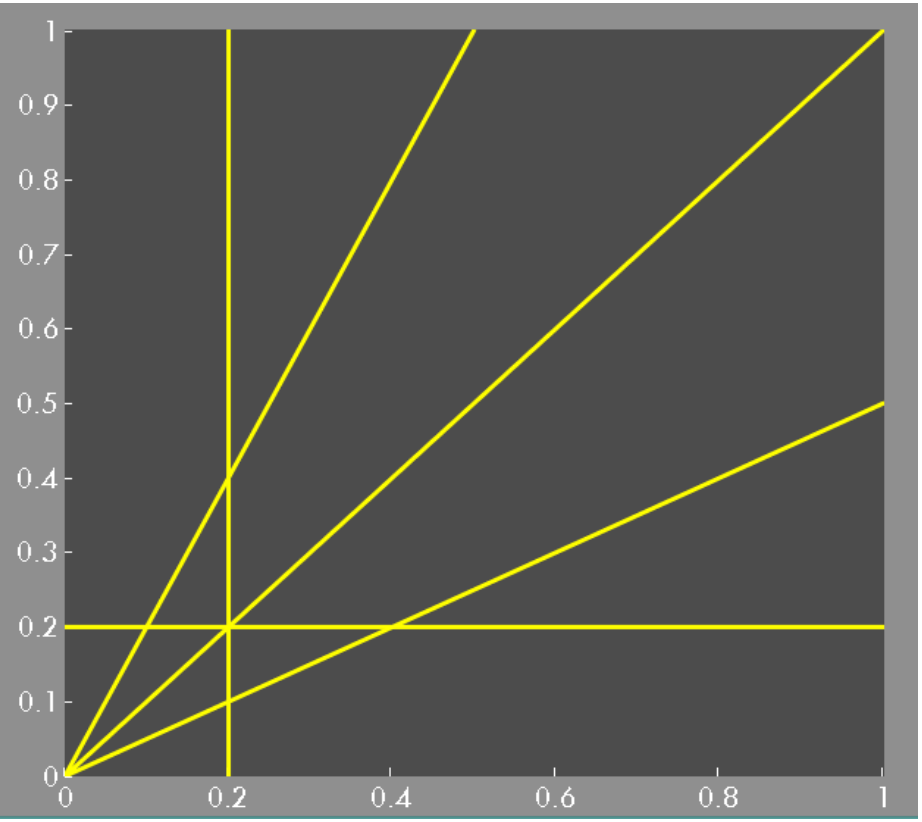
Bitmap Vector Graph

MATLAB Line smoothing

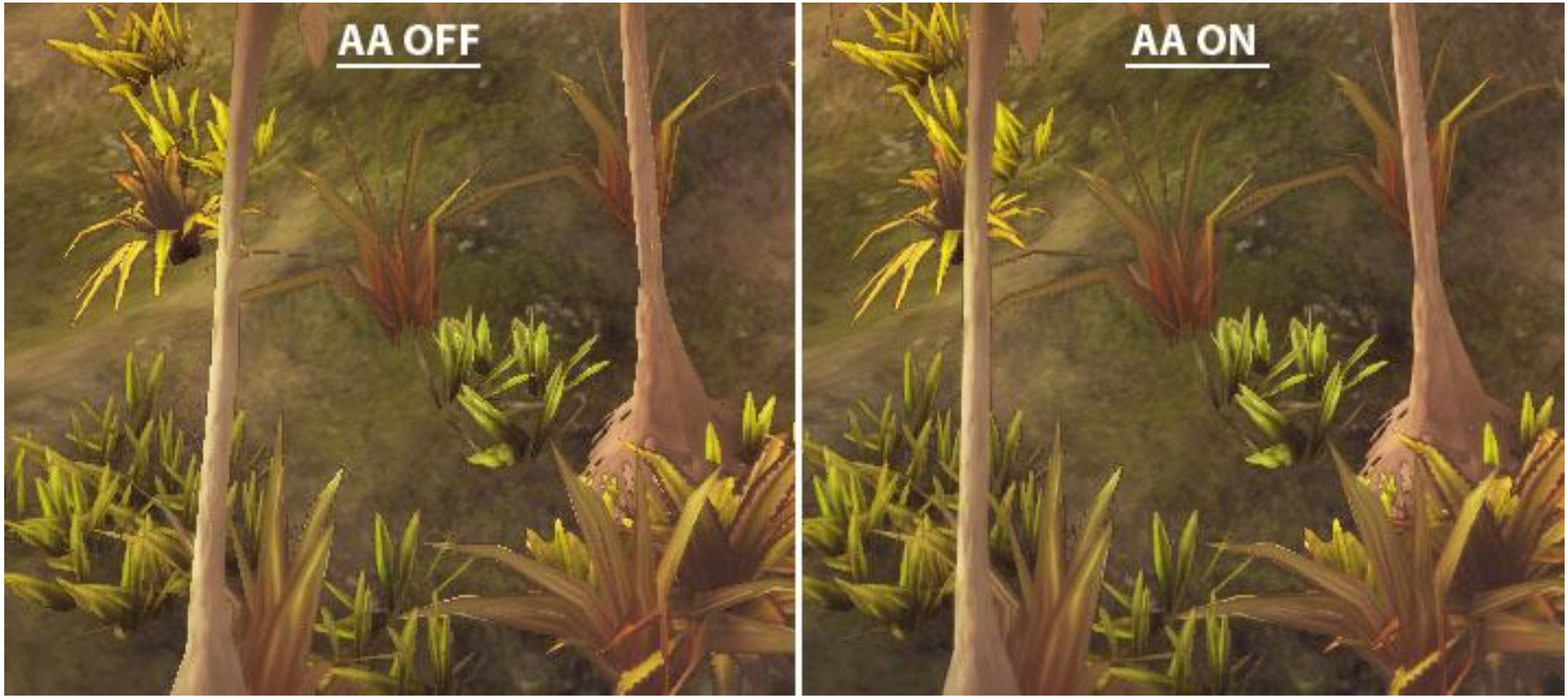
Ex : plot([0, 1], [0, 1], 'LineSmoothing', 'off');



Ex : plot([0, 1], [0, 1], 'LineSmoothing','on');



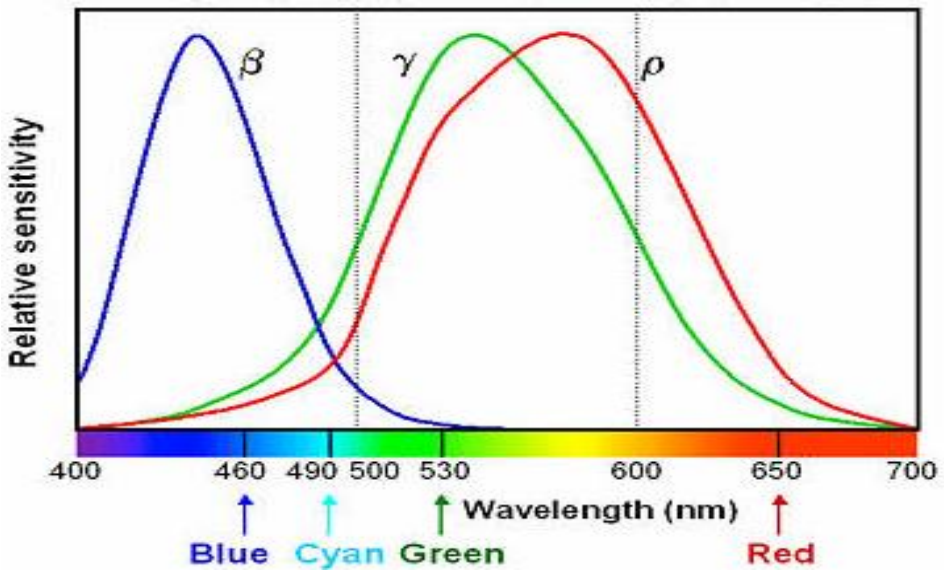
Aliasing V.S. Anti-aliasing



http://solidlystated.com/software/from-dust-anti-aliasing/

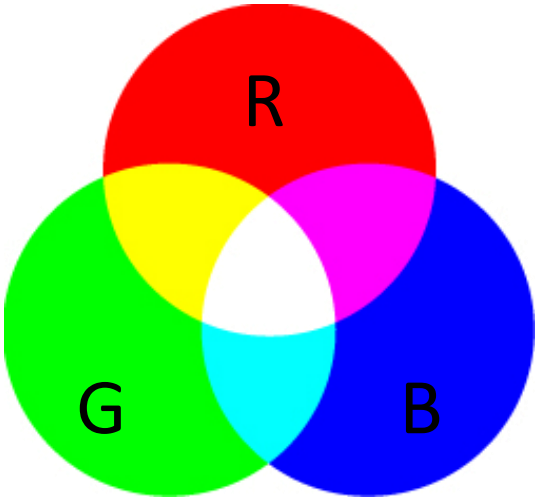
Color

- Color is composed of electromagnetic waves
  - The wavelength of visible color fall between approximately 370 and 780 nanometers.
- Three Characteristics of colors
  - Hue (essential color): *dominant wavelength*
  - Saturation (color purity)
  - Luminance (lightness)
- Color model
  - RGB color model
  - CMY color model
  - HSV color model
  - YIQ color model



RGB color mode

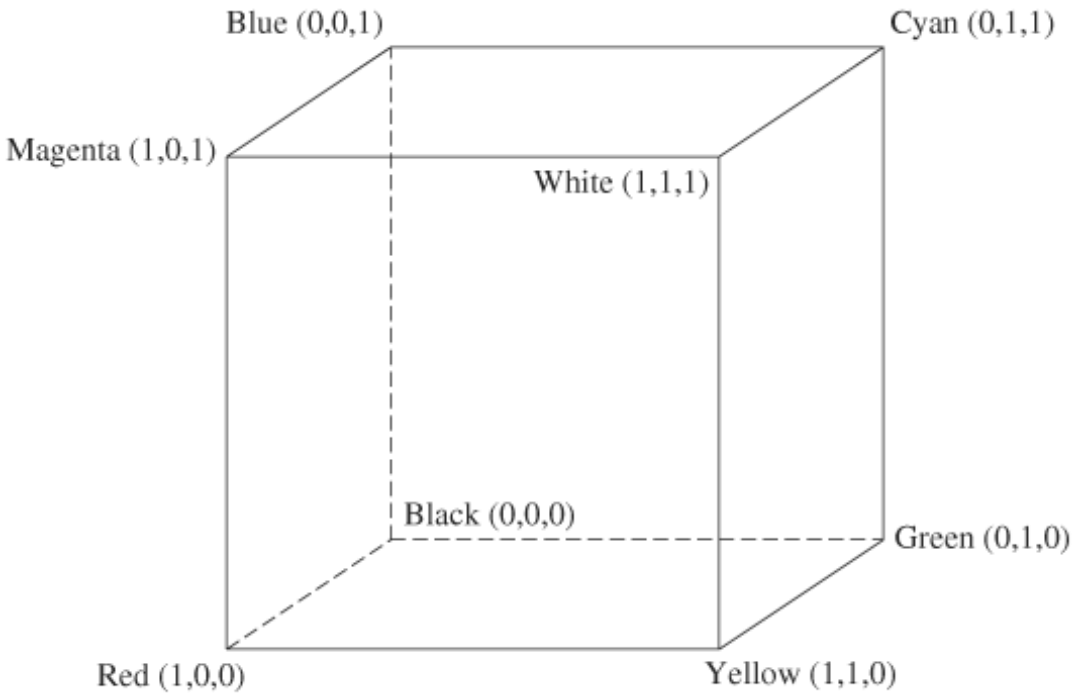
- Additive color mixing
- Combinations of three primary colors. No one of them can be created as a combination of the other two
  - Red
  - Green
  - Blue



- C = *r*R + *g*G + *b*B
  - *R*, *G*, *B*: constant values based on the wavelengths
  - *r*, *g*, *b*: the relative amounts. The values r, g, b are referred to as the values of the **RGB color components** (color channels)

RGB color cube

- R,G, and B correspond to three axes in 3D space
- Normalize the relative amount of R,G and B in a color. Each value varies between 0 and 1.



RGB color to grayscale

- The conversion of RGB color to grayscale
  - In RGB color (R, G, B)
  - In grayscale (L, L, L)

L = 0.299R + 0.587G + 0.114B

- Since all three color components are equal in a gray pixel, only one of the three values needs to be stored. Thus a 24-bit RGB pixel can be stored as an 8-bit grayscale pixel.

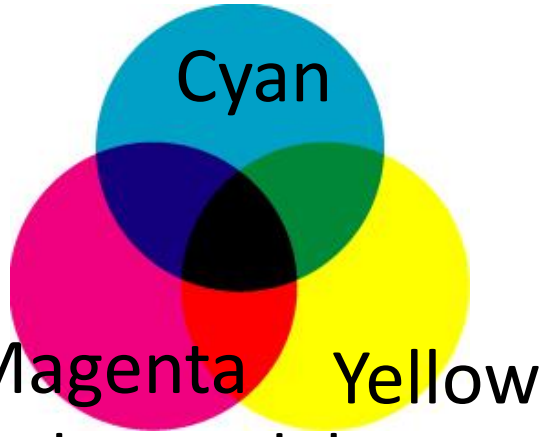
Image from google

RGB color model – advantage & disadvantage

- Advantage
  - The human eye perceives color
  - The computer monitor can be engineered to display color
- Disadvantage
  - There exist visible colors that can't be represented with positive value for each of the red, green and blue components
  - It is necessary to "subtract out" some of the red, green, or blue in the combined beams to match the pure color

CMY color model

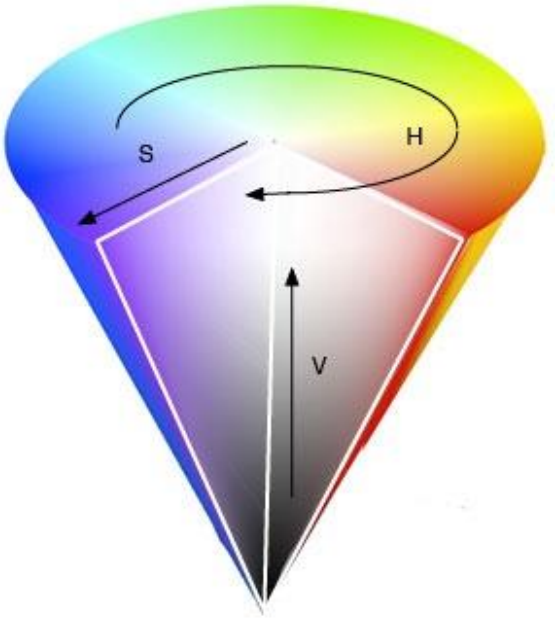
- Subtractive color mixing
- Commonly used in printing
- It has three primaries
  - Cyan, Magenta, Yellow



- Conversion between RGB and CMY color models
  - C = 1 – R, M = 1-G, Y = 1 – B
- CMYK model
  - In CMY color mode, the maximum amount of three primary should combine to black, but in fact producing a dark brown
  - Add a pure black -> K
  - K = min(C, M, Y), $C_{new} = C - K$, $M_{new} = M - K$, $Y_{new} = Y - K$

HSV color model

- Represent color by three terms
  - **Hue**, the color type (such as red, blue, or yellow)
    - ✓Ranges from 0-360 (but normalized to 0-100% in some applications)
  - **Saturation**, the "vibrancy" of the color
    - ✓Ranges from 0-100%
    - ✓The lower the saturation of a color, the more “grayness”
  - **Value**, the brightness of the color
    - ✓Ranges from 0-100%



- Also called HSB(hue, saturation and brightness)

ALGORITHM 2.3 - RGB TO HSV

```
/* Input: r, g, and b, each real numbers in the range [0 . . . 1].
Output: h, a real number in the range of [0 . . . 360),
except if s = 0, in which case h is undefined.
s and v are real numbers in the range of [0 . . . 1].*/
{
  max = maximum(r,g,b)
  min = minimum(r,g,b)
  v = max
  if max ≠ 0 then s = (max − min)/max
  else s = 0
  if s == 0 then h = undefined
  else {
    diff = max − min
    if r == max then h = (g − b) / diff
    else if g == max then h = 2 + (b − r) / diff
    else if b == max then h = 4 + (r − g) / diff
    h = h * 60
    if h < 0 then h = h + 360
  }
}
```

YIQ color model

- It captures the luminance information in one value and puts the color (chrominance) information in the other two values
  - Luminance component: Y
  - Chrominance components: I, Q
- Convert RGB to YIQ

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- Translate YIQ to RGB → Invert the matrix
- YIQ color model is used in U.S. commercial television broadcasting
- Advantage in color/black and white broadcasting

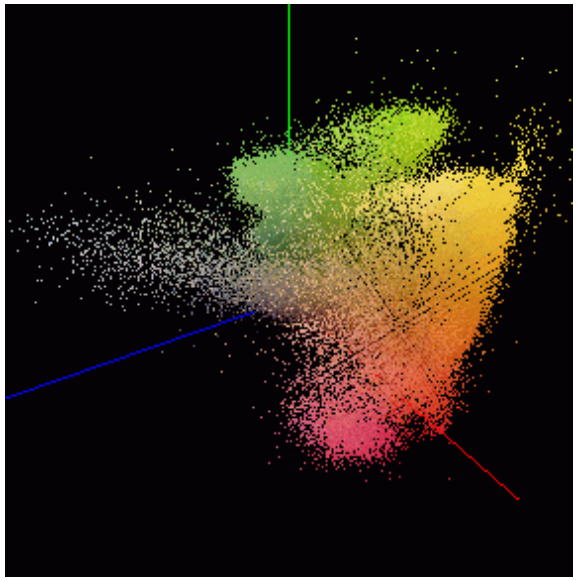
Color Quantization

- Color quantization begins with an image file stored with a bit depth of n and reduces the bit depth to b.
- The number of colors representable in the original file is 2^n, and the number of colors representable in the adjusted file will be 2^b.
- As an example, let's assume your image is initially in RGB mode with 24-bit color, and you want to reduce it to 8-bit color.

Process of Color Quantization

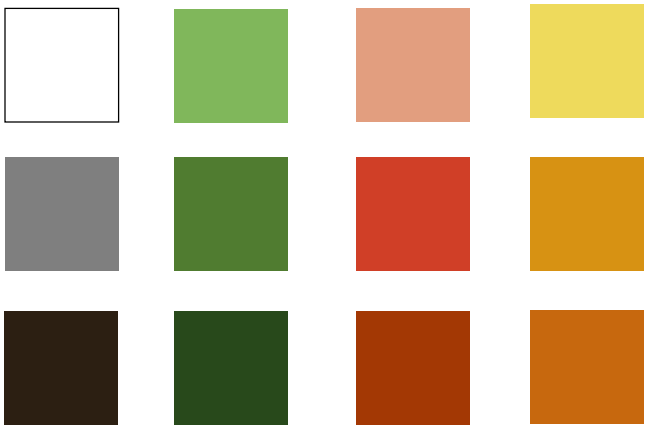
- The process of color quantization involves three steps

1. The actual range and number of colors used in your picture must be determined. If your image is stored initially in RGB mode with 24-bit color, then there are 2^{24} = 16,777,216 possible colors. The question is, ***which of these colors appear in the picture?***



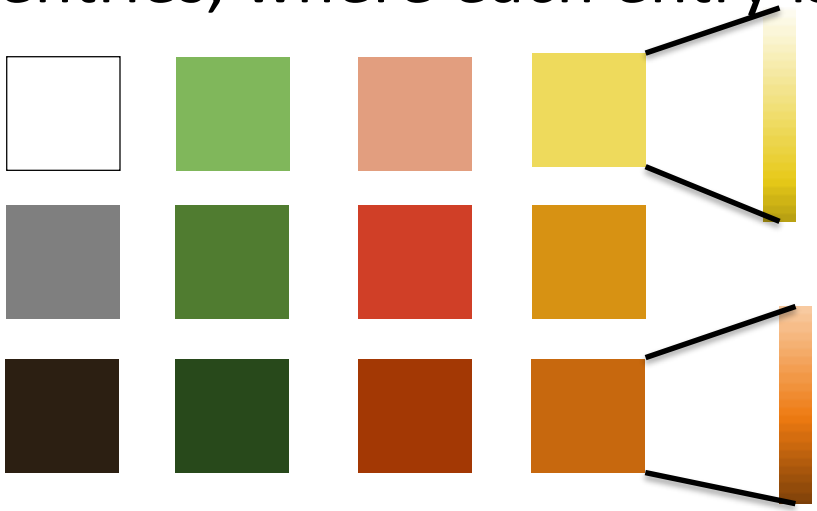
Process of Color Quantization

2. The second step entails choosing 2^b colors to represent those that actually appear in the picture. For our example, the adjusted picture would be limited to 2^8 = 256 colors



Process of Color Quantization

3. To map the colors in the original picture to the colors chosen for the reduced bit-depth picture. The *b* bits that represent each pixel then become an index into a color table that has 2^b entries, where each entry is *n* bits long. In our example, the table would have 256 entries, where each entry is 24 bits long.



Popularity algorithm

- One simple way to achieve a reduction from a bit depth of *n* to a bit depth of *b*.

1. The 2^b colors that appear most often in the picture are chosen for the reduced-bit depth picture
2. To map one of the original colors to the more limited palette is by finding the color that is most similar using the minimum mean squared distance.

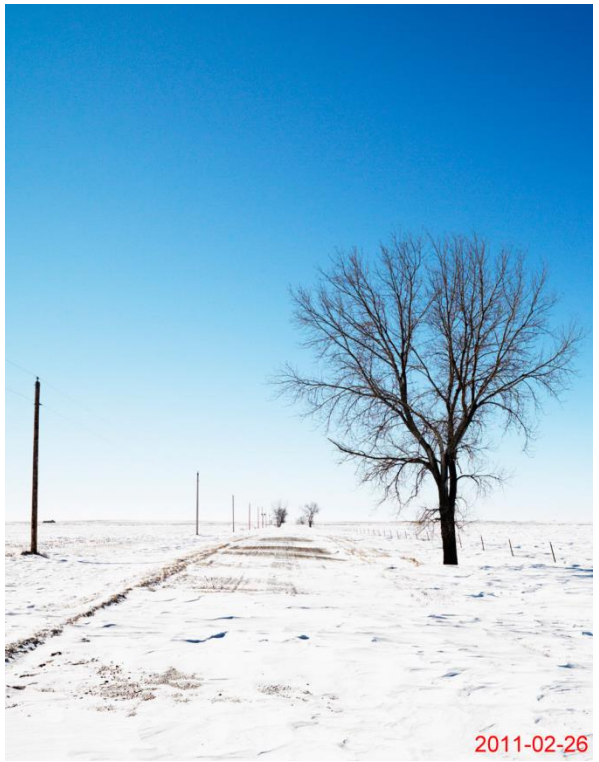
   → $\min((R - r_i)^2 + (G - g_i)^2 + (B - b_i)^2)$

   R, G, B: original color

   r_i, g_i, b_i: quantized color

Popularity algorithm - Disadvantage

- Disadvantage: it completely throws out colors that appear infrequently
  - Example:
    A picture with one dramatic spot of red in a field of white snow, trees, and sky may lose the red spot entirely, completely changing the desired effect.



Color Quantization

Original image

24 bit image

Quantized image

4 bit image

3 bit image

2 bit image

Color Quantization

Original image

24 bit image

Quantized image

4 bit image

3 bit image

2 bit image

Dithering

- ***Dithering*** is a technique for simulating colors that are unavailable in a palette by using available colors that are blended by the eye so that they look like the desired colors
- Dithering is helpful when you change an image from RGB mode to indexed color because it makes it possible to reduce the bit depth of the image, and thus the file size, without greatly changing the appearance of the original image

Noise Dithering

- Also called ***random dithering***

1. Generate a random number from 0 to 255
2. If the pixel's color value is greater than the number then it is white, otherwise black
3. Repeat step 2 for each pixel in the image

- Crude and "noisy"

Example of Noise Dithering

Average Dithering

1. Calculate an average pixel value
2. If a pixel value is above this average, then set it to white, else black
3. Repeat step 2 for each pixel in the image

- Crude and “contrasty”

Example of Average Dithering

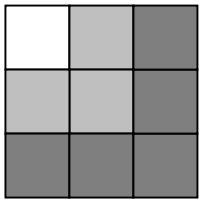
Pattern Dithering

- Also called ***ordered dithering***, ***Bayer method***

1. Break the image into small blocks
2. Define a *threshold matrix*
   - Use a different threshold for each pixel of the block
   - Compare each pixel to its own threshold

- Widely used by the printing industry - rare in multimedia

Image Block

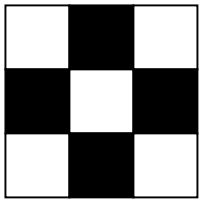


$$\begin{bmatrix} 255 & 192 & 128 \\ 192 & 192 & 128 \\ 128 & 128 & 128 \end{bmatrix}$$

Threshold matrix

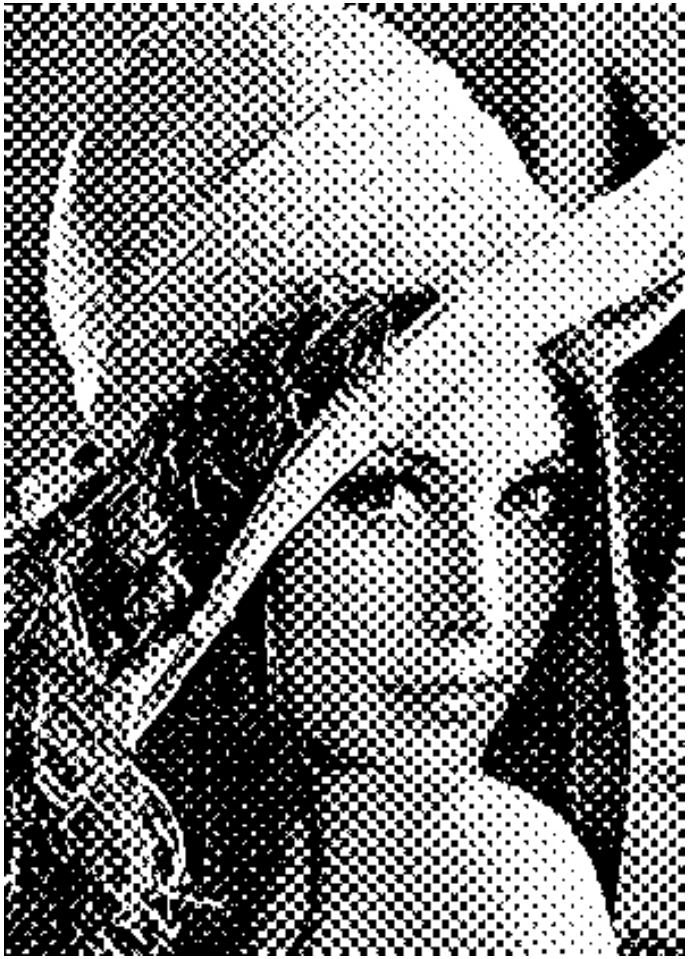
$$\begin{bmatrix} 200 & 250 & 100 \\ 220 & 150 & 200 \\ 10 & 150 & 50 \end{bmatrix}$$

Result



$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Pattern Dithering



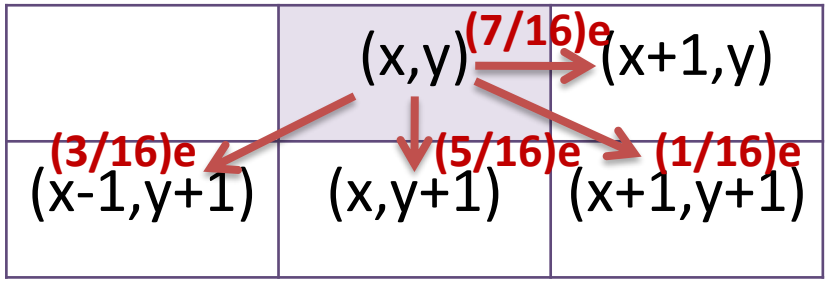
Error Diffusion Dithering

- Also called ***Floyd-Steinberg algorithm***
- Disperses the error or difference between a pixel's original value and the color (or grayscale) value available

Error Diffusion Dithering

1. Define the mask, for example:
2. For each pixel p
   1. Define e: if p<128, e = p; otherwise e = p-255
   2. Change the value of neighbor pixel

	p	7
3	5	1



p(x+1,y) = p(x+1,y)+(7/16)e
p(x-1,y+1) = p(x-1,y+1)+(3/16)e
p(x,y+1) = p(x,y+1)+(5/16)e
p(x+1,y+1) = p(x+1,y+1)+(1/16)e

- When the mask is moved to the right by one pixel, the next step will operate on a pixel that has possibly been changed in a previous step

Error Diffusion Dithering

- After the error has been distributed over the whole image, the pixels are processed a second time. This time for each pixel, if the pixel value is less than 128, it is changed to a 0 in the dithered image. Otherwise it is changed to a 1.

After dithering

Image Transform

- An **image transform** is a process of changing the color or grayscale values of image pixels
- Image transforms can be divided into two types
  - ***Pixel point processing***
    a pixel value is changed based only on its original value, without reference to surrounding pixels
  - ***Spatial filtering***
    changes a pixel's value based on the values of neighboring pixels

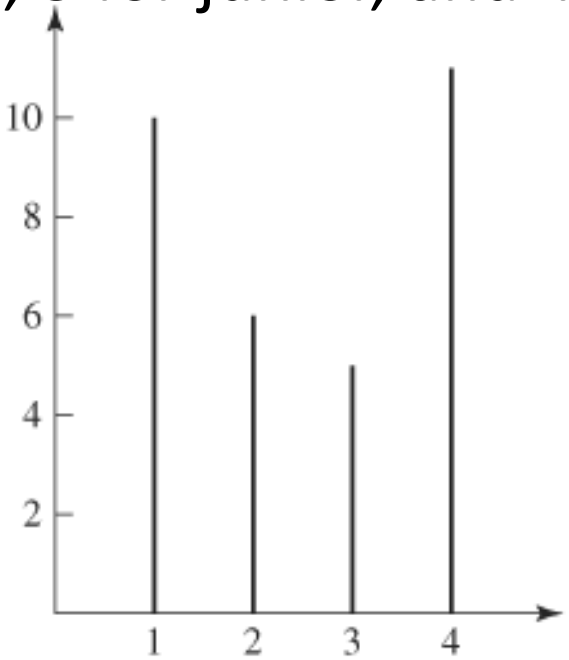
Histogram

- A ***histogram*** is a discrete function that describes frequency distribution; that is, it maps a range of discrete values to the number of instances of each value in a group of numbers.
- Let v_i be the number of instances of value i in the set of numbers. Let *min* be the minimum allowable value of i, and let *max* be the maximum. Then the ***histogram function*** is defined as

$$h(i) = v_i \text{ for } min \leq i \leq max.$$

Histogram

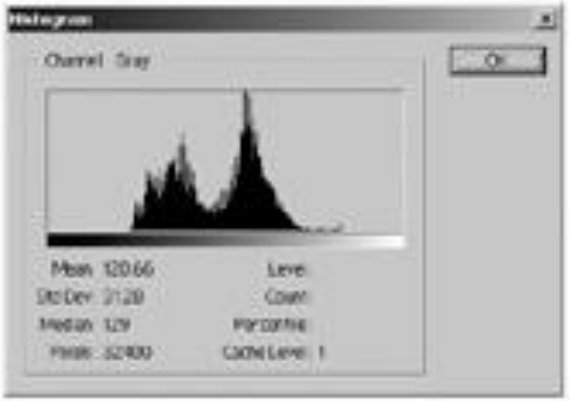
- Simple histogram
  a group of 32 students identified by class (1 for freshman, 2 for sophomore, 3 for junior, and 4 for senior)

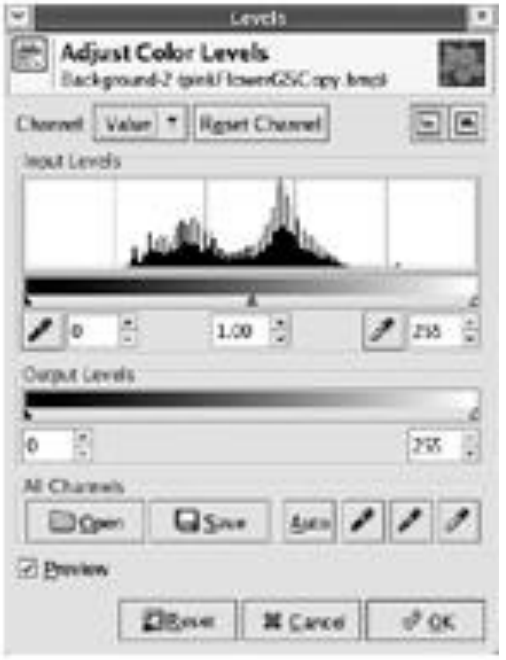


- There are ten freshmen, six sophomores, five juniors, and 11 seniors

Histogram

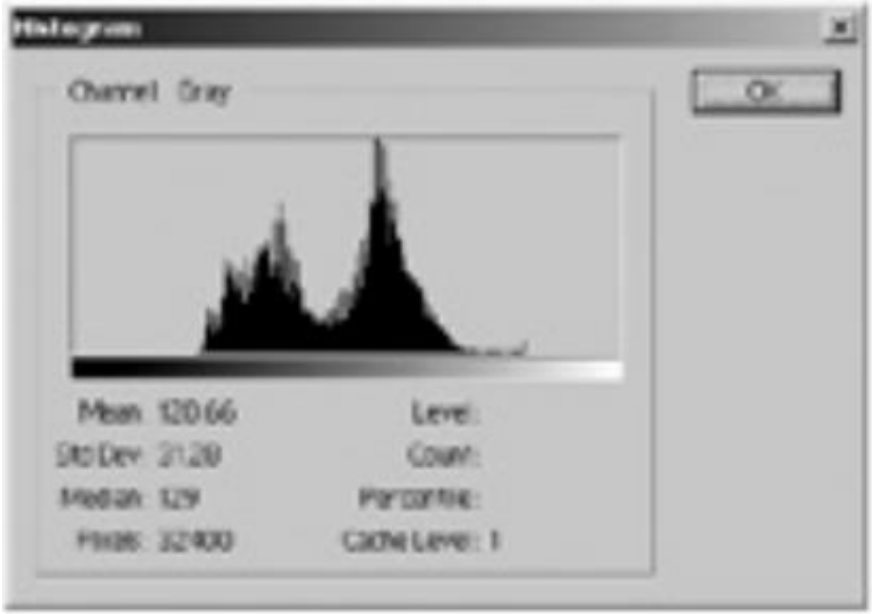
- An image histogram maps pixel values to the number of instances of each value in the image





Histogram

- **Mean or average**
  - $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
- **Median**
  - A ***median*** is a value x such that at most half of the values in the sample population are less than x and at most half are greater.



- **Standard deviation**
  - $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(xi - \overline{x})^2}$

A large standard deviation implies that most of the pixel values are relatively far from the average, so the values are pretty well spread out over the possible range

Luminance histogram

- Some scanners or image processing programs give you access to a ***luminance histogram*** (also called a ***luminosity histogram***) corresponding to a color image

$$L = 0.299R+0.587G+0.114B$$

- Among the three color channels, the human eye is most sensitive to green and least sensitive to blue

Example of Histograms

A histogram that can be manipulated to adjust brightness and contrast (from Photoshop)

Histogram of image after contrast adjustment (from Photoshop)

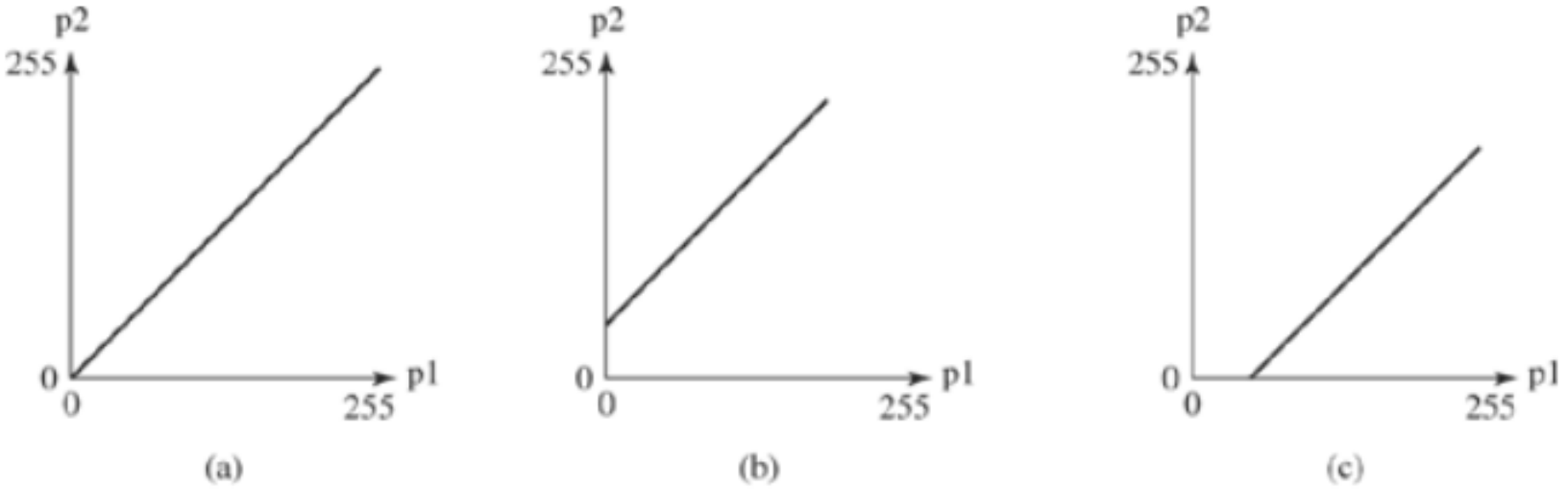
Transformation Function

- In programs like Photoshop and GIMP, the Curves feature allows you to think of the changes you make to pixel values as a transform function
- We define a ***transform*** as a function that changes pixel values

$$g(x, y) = T(f(x, y)) \text{ , } p_2 = T(p_1)$$

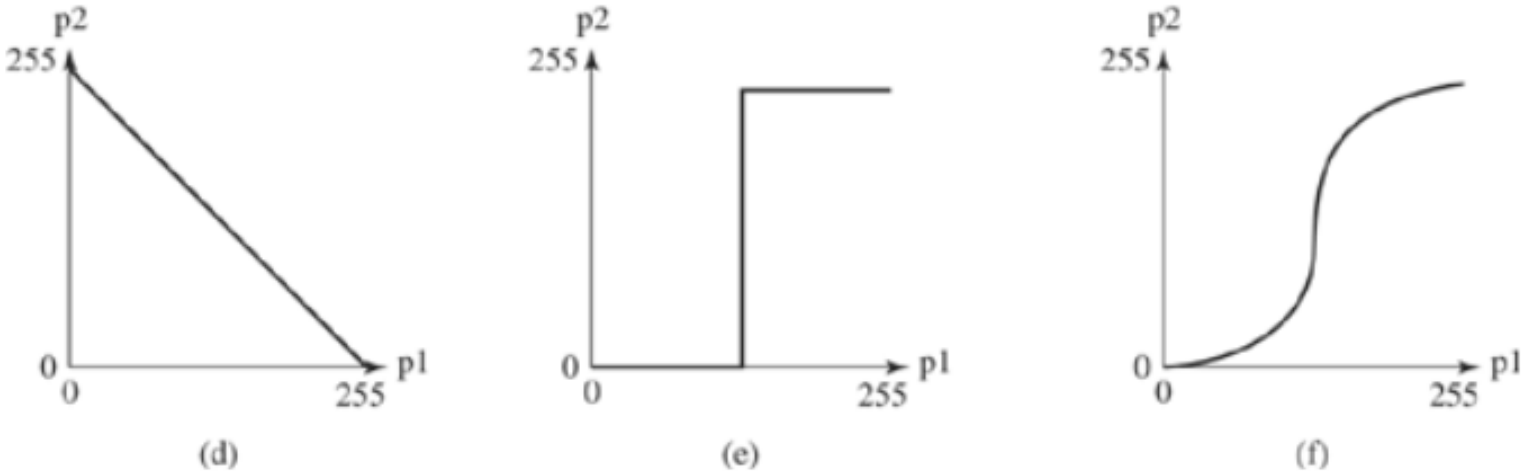
  - $f(x, y)$ is the pixel value at that position (x, y) in the original image. Abbreviate $f(x,y)$ as p_1
  - T is the transformation function
  - $g(x, y)$ is the transformed pixel value. Abbreviate $f(x,y)$ as p_2

Curve Representation



a) The transform doesn't change the pixel values. The output equals the input.
b) The transform lightens all the pixels in the image by a constant amount.
c) The transform darkens all the pixels in the image by a constant amount.

Curve Representation



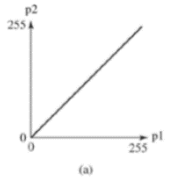
d) The transform inverts the image, reversing dark pixels for light ones.

e) The transform is a threshold function, which makes all the pixels either black or white. A pixel with a value below 128 becomes black, and all the rest become white.

f) The transform increases contrast. Darks become darker and lights become lighter.

Intensity Transformation

- Adjusting contrast and brightness with curves function

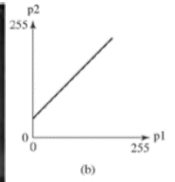


(a)

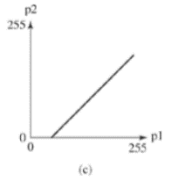

(a) Original image


(b) Lighten

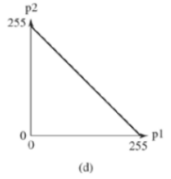

(c) Darken



(b)



(c)

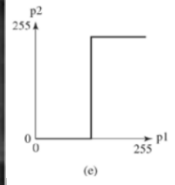


(d)

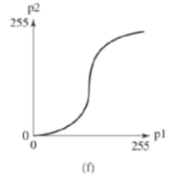

(d) Invert


(e) Threshold


(f) Increase contrast



(e)



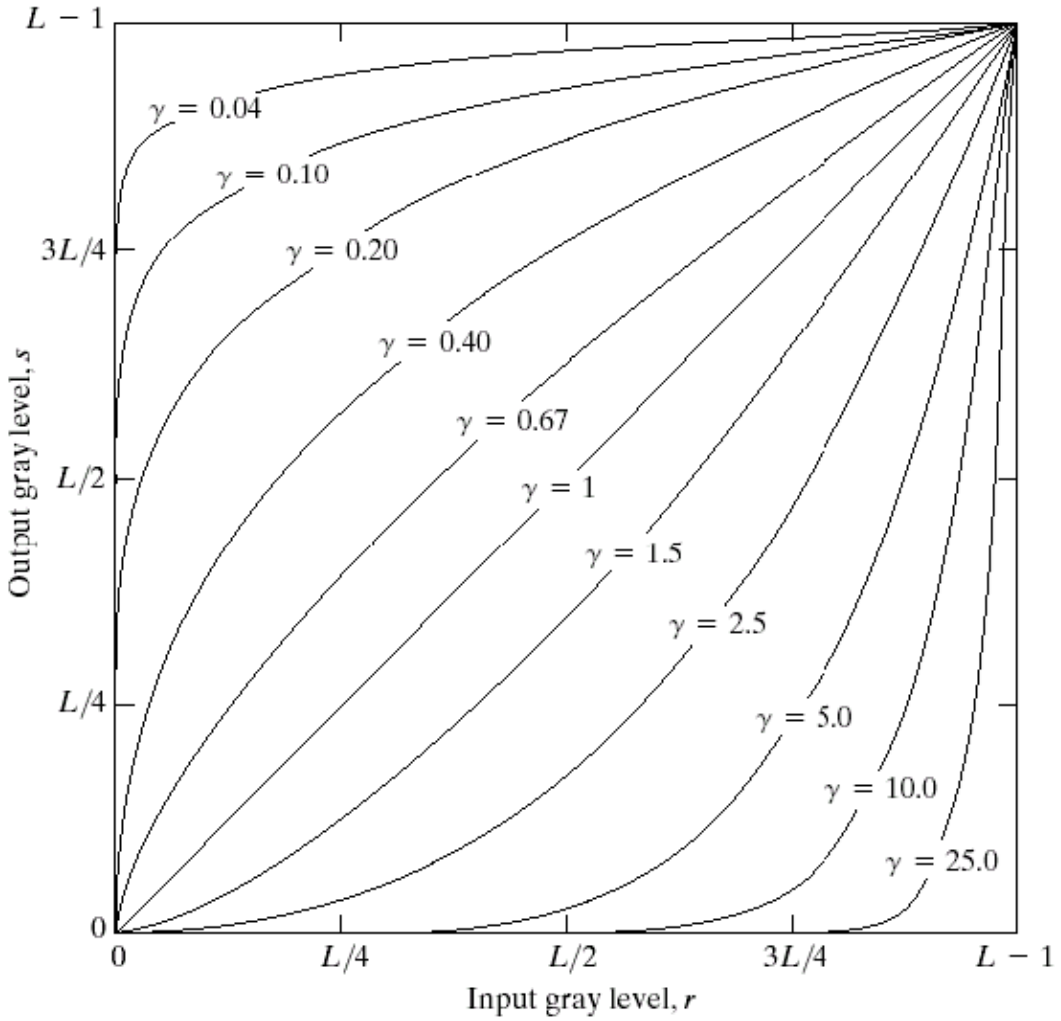
(f)

Gamma Transform

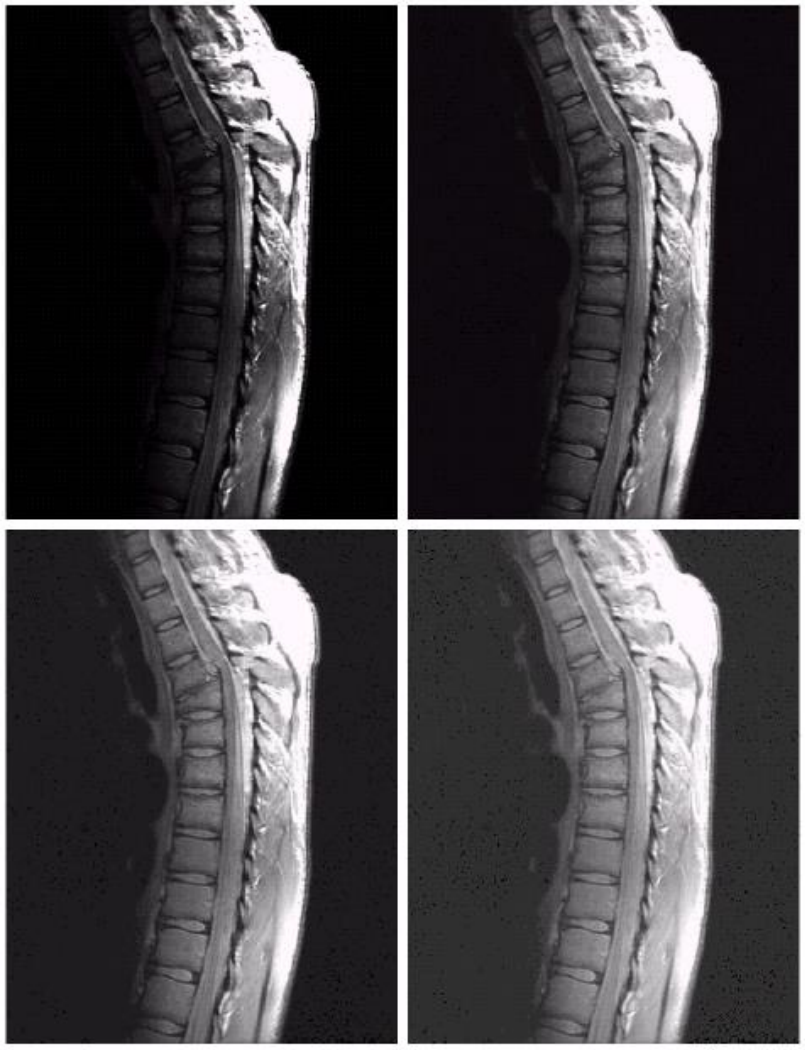
- The ***gamma value*** γ is an exponent that defines a nonlinear relationship between the input level *r* and the output level *s* for a pixel transform function.

$$s = r^{\gamma}, 0 \leq s \leq 1$$

Gamma Transform



Gamma Transform -Examples



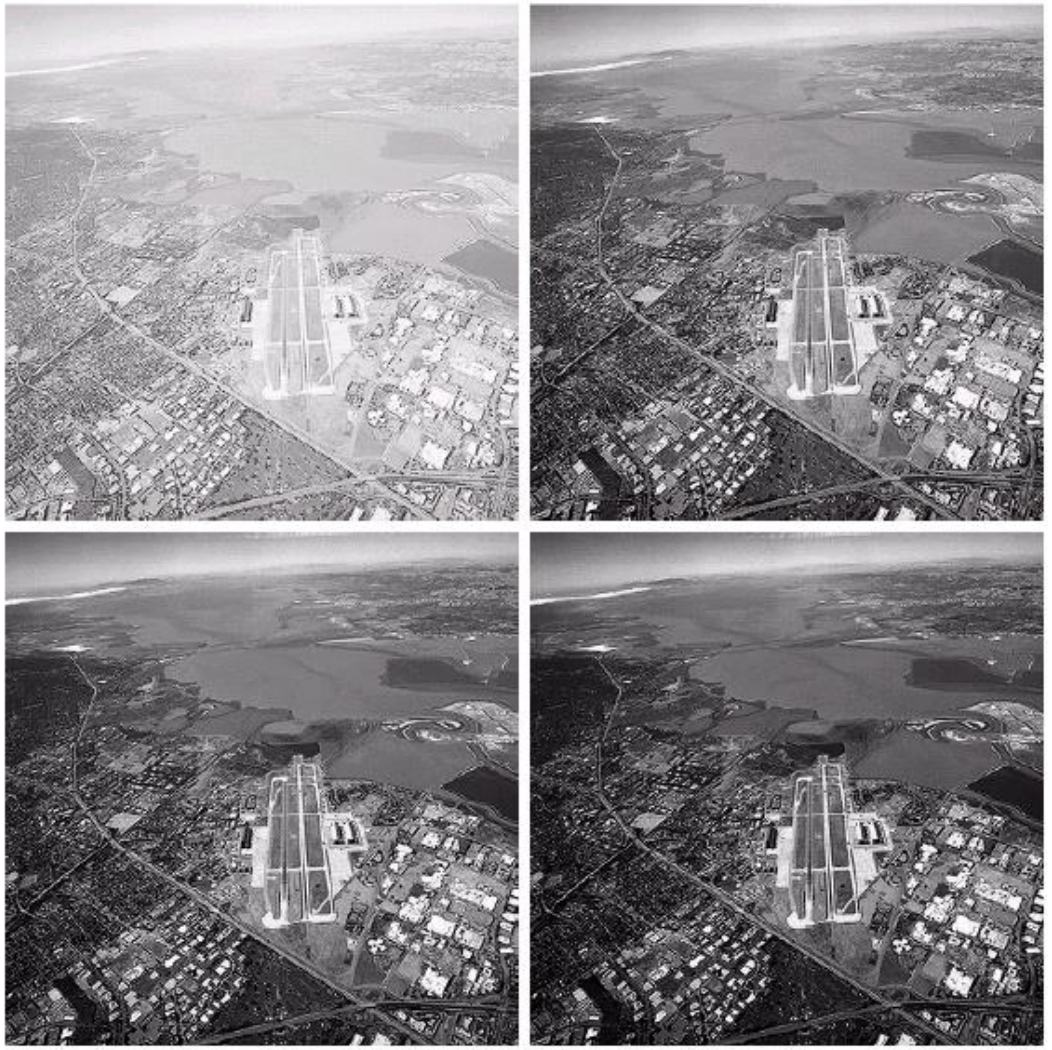
a b
c d

FIGURE 3.8
(a) Magnetic resonance (MR) image of a fractured human spine.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3, respectively. (Original image for this example courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Gamma Transform - Examples

a b
c d

FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0, respectively. (Original image for this example courtesy of NASA.)



Gamma Transform -Examples

Original image

Gamma = 3

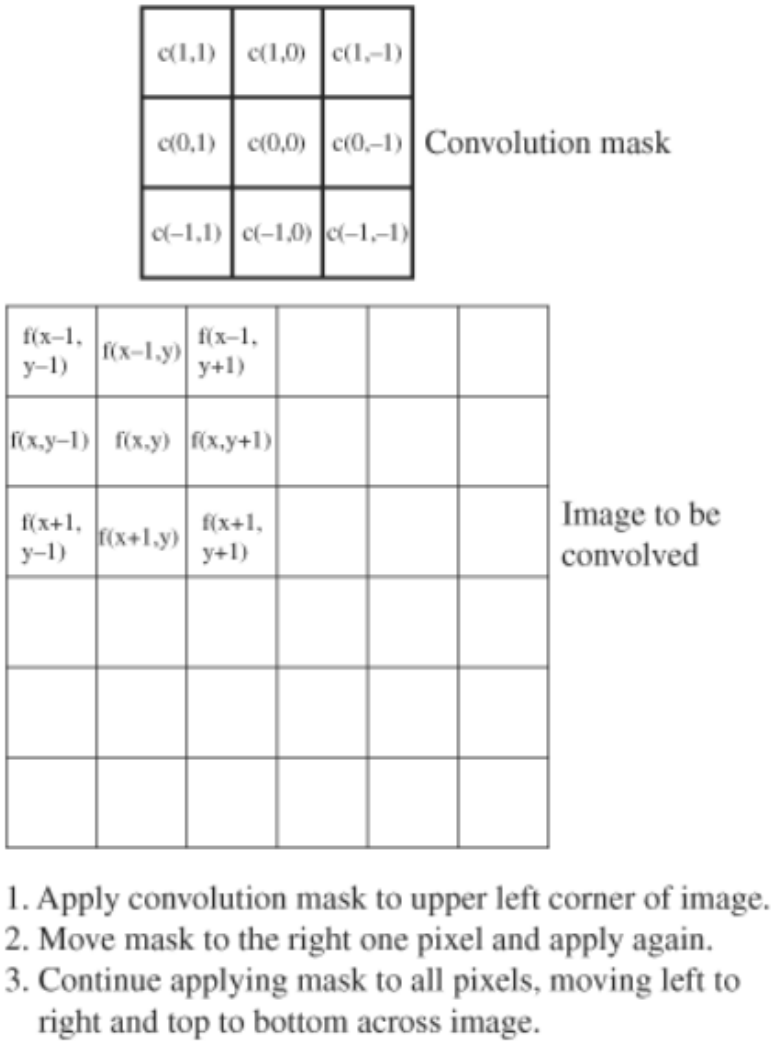
Gamma = 0.5

Filter

- A ***filter*** is an operation performed on digital image data to sharpen, smooth, or enhance some feature, or in some other way modify the image
- ***Filtering in the frequency domain*** is performed on image data that is represented in terms of its frequency components
- ***Filtering in the spatial domain*** is performed on image data in the form of the pixel's color values

Convolution

- Spatial filtering is done by a mathematical operation called ***convolution***, where each output pixel is computed as a weighted sum of neighboring input pixels
- Convolution is based on a matrix of coefficients called a ***convolution mask***. The mask is also sometimes called a ***filter***.



Convolution

- Let $f(x, y)$ be an $M \times N$ image and $c(v, w)$ be an $m \times n$ mask. Then the equation for a linear convolution is

$$g(x, y) = \sum_{v=-i}^{i} \sum_{w=-j}^{j} c(v, w) f(x - v, y - w)$$

where $i = (m - 1)/2$ and $j = (n - 1)/2$. Assume m and n are odd. This equation is applied to each pixel $f(x, y)$ of an image, for $0 \le x \le M - 1$ and $0 \le y \le N - 1$.
(If $x - v < 0$, $x - v \ge M$, $y - w < 0$, or $y - w \ge N$, then $f(x, y)$ is undefined. These are edge cases, discussed below.)

Convolution for averaging pixels

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Convolution mask

202	232	222	222	221	221
202	202	212	200	199	202
202	222	192	199	180	188
202	227	201	193	185	178
200	196	202	189	180	173
201	190	188	182	181	174

Image to be convolved

Move the convolution mask over an area of pixels in the original image.

1/9 202	1/9 232	1/9 222	222	221	221
1/9 202	1/9 202	1/9 214	200	199	202
1/9 202	1/9 199	1/9 193	199	180	188
202	227	201	193	185	178
200	196	202	189	180	173
201	190	188	182	181	174

This mask will compute an average of the pixels in the neighborhood. The value of the center pixel will become

1/9*202+1/9*232+1/9*222+1/9*202+1/9*202+
1/9*214+1/9*202+1/9*199+1/9*193

Handling edges in convolution

1/9	1/9	1/9				
1/9	1/9 202	1/9 232	222	222	221	221
1/9	1/9 202	1/9 202	214	200	199	202
	202	199	193	199	180	188
	202	227	201	193	185	178
	200	196	202	189	180	173
	201	190	188	182	181	174

Four ways to convolve pixels at the edge of an image:

1. Assume that there are zero-valued pixels around the edges. These would be under the portion of the mask shaded in gray;
   or
2. Replicate the values from the edges, as shown; →

1/9 202	1/9 202	1/9 232
1/9 202	1/9 202	1/9 232
1/9 202	1/9 202	1/9 202

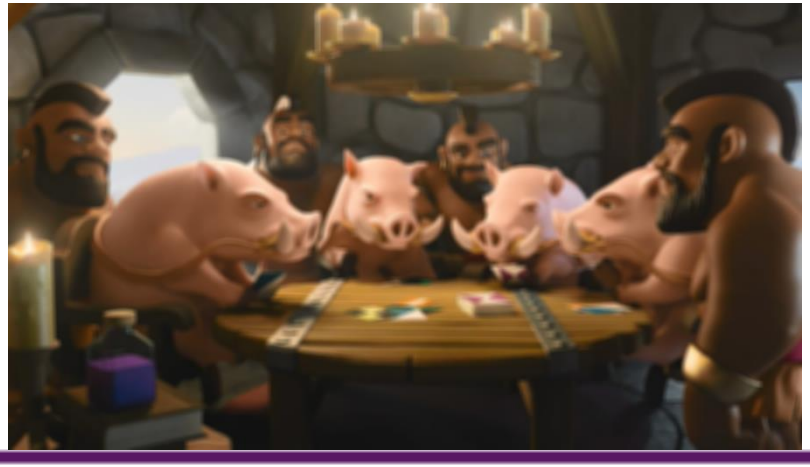
   or
3. Use only the portion of the mask covering the image and change the weights appropriately for that step;
   or
4. Don't do convolution on the pixels at the edges.

Average filter

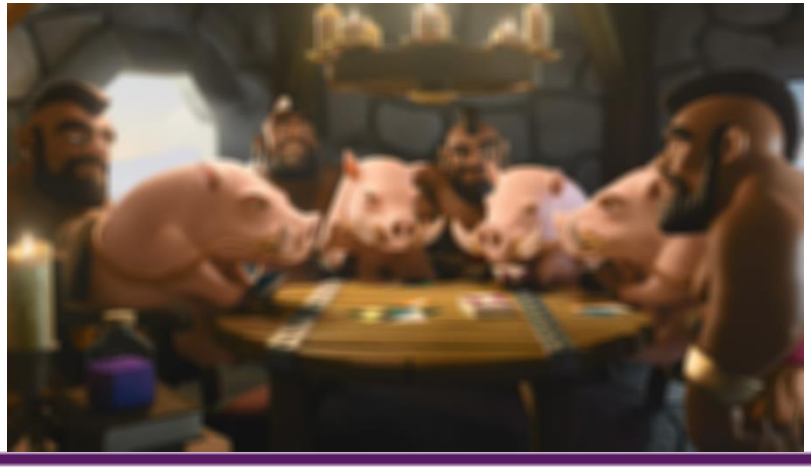
- Output pixel is the mean of its kernel neighbors.
- Here shows the effects of average filters with different mask size.

Original image

Mask size: 15



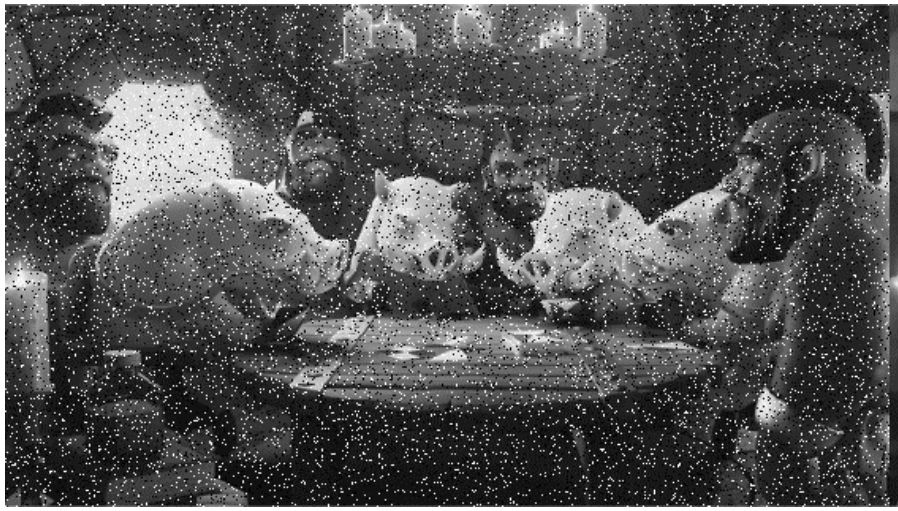
Mask size: 25



Median filter

- Output pixel is the median of its neighboring pixels in mask
- Median filter is able to perform salt-and-pepper noise reduction.

Original image



After applying median filter

Unsharp mask

- The name is misleading because this filter actually sharpens images
- The pixel values in the original image are doubled, and the blurred version of the image is subtracted from this

													−1	
	2		−	1	−3	1	=	−1	5	−1
					1						−1	

2*original − Blur mask = Unsharp mask

Original image

Image with blur filter applied

Image with unsharp mask applied

Edge-detection filter

- Sobel filter detects the edge, making that edge white while everything else is black

1	1	1
0	0	0
–1	–1	–1

Mask

255	255	255	255
255	255	255	255
255	255	255	255
0	0	0	0
0	0	0	0
0	0	0	0

Block **a**

0	0	0	0
0	0	0	0
255	255	255	255
255	255	255	255
0	0	0	0
0	0	0	0

Block **b**

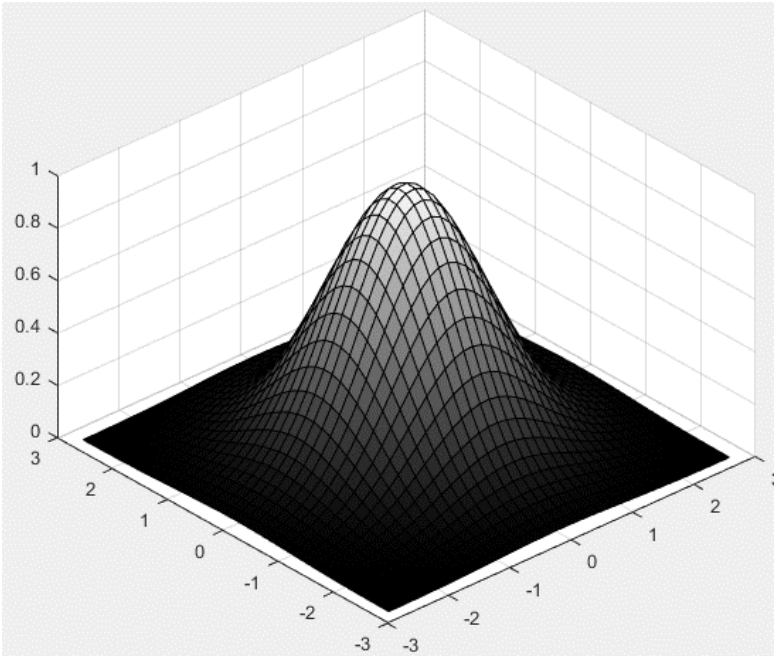
The mask above applied to block **a** of pixels yields block **b**.

Gaussian filter

- An alternative for smoothing is to use a ***Gaussian filter***, where the coefficients in the convolution mask get smaller as you move away from the center of the mask

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

Gaussian convolution mask

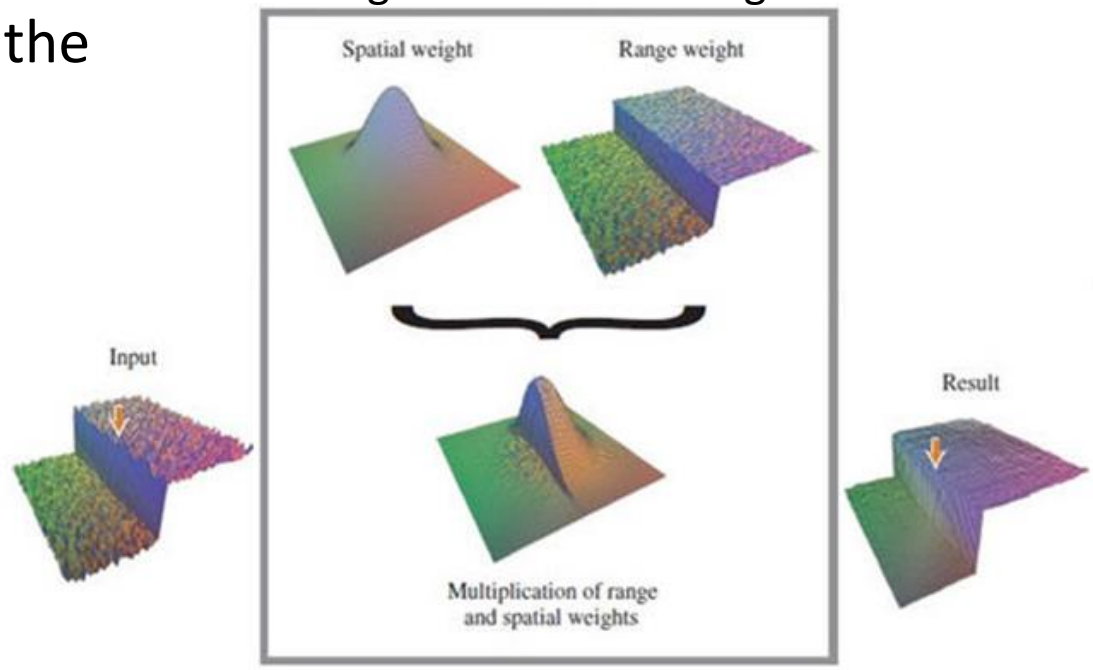


Bilateral filter

- Comparing to Gaussian filter, the bilateral filter considers the mask weights with two components:
  - Gaussian filter mask.
  - Intensity difference between neighboring pixels and the evaluated one.

Origin/filtered

Edge-aware smoothing

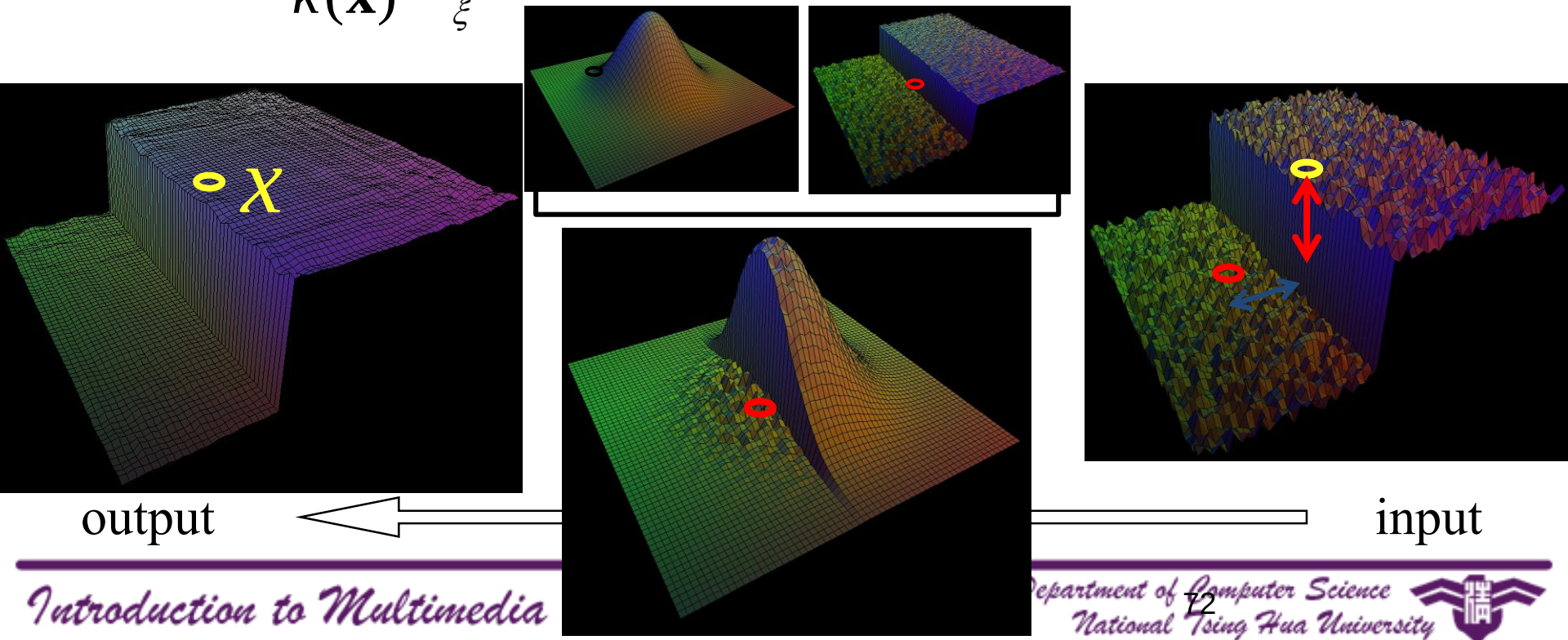


Bilateral filtering

[Tomasi and Manduchi 1998]

- Spatial Gaussian h
- Gaussian w on the intensity difference

$$g(\mathbf{x}) = \frac{1}{k(\mathbf{x})} \sum_{\xi} h(\mathbf{x},\xi)\ w(f(\xi) - f(\mathbf{x}))\ f(\xi)$$



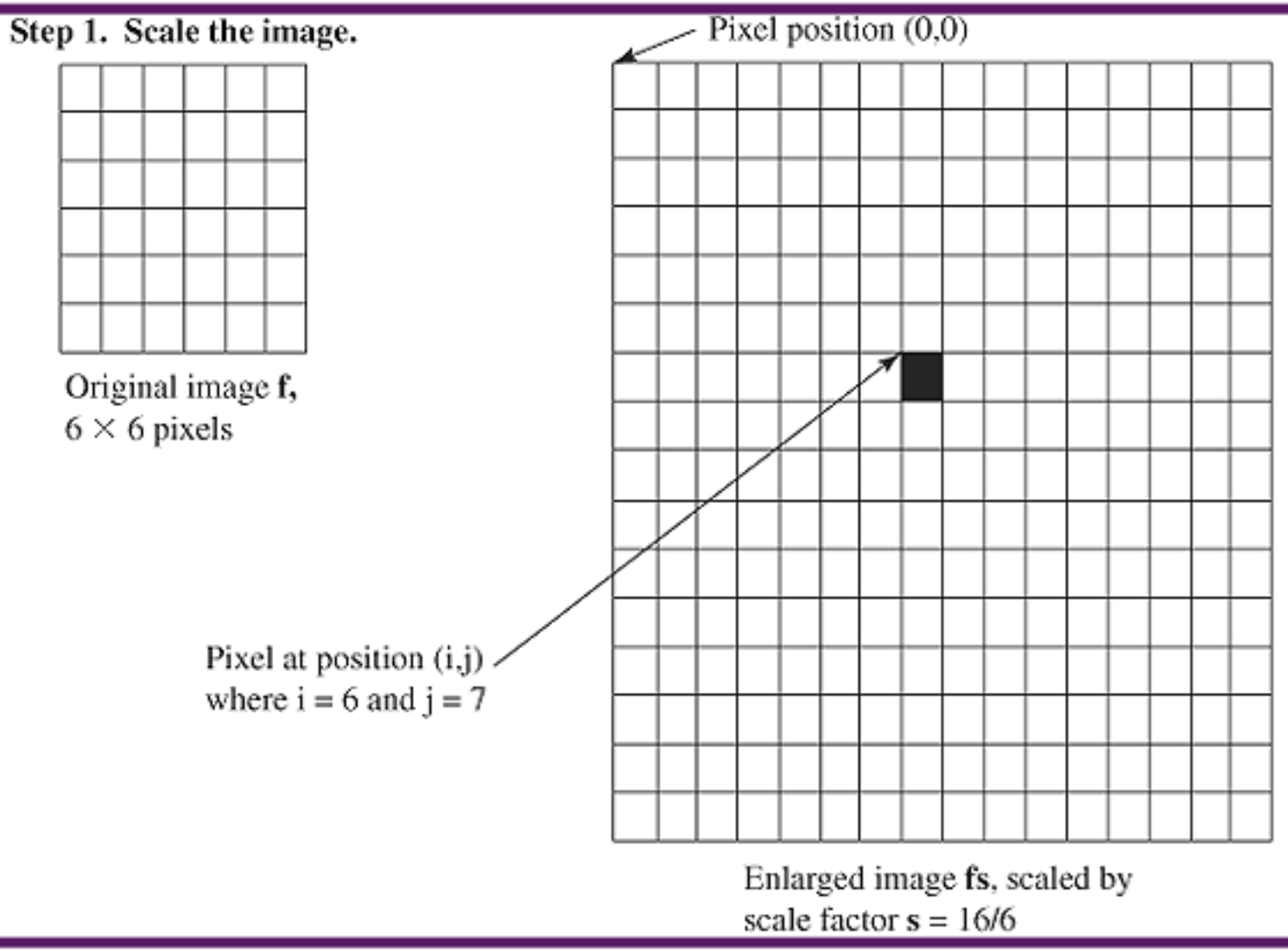
Example of Bilateral Filtering

From B. Weiss, “Fast median and bilateral filtering”, Proc. ACM SIGGRAPH, 2006.

Interpolation

- There are interpolation methods for resampling that give better results than simple replication or discarding of pixels
- ***Interpolation*** is a process of estimating the color of a pixel based on the colors of neighboring pixels
  - Nearest neighbor
  - Bilinear
  - Bicubic

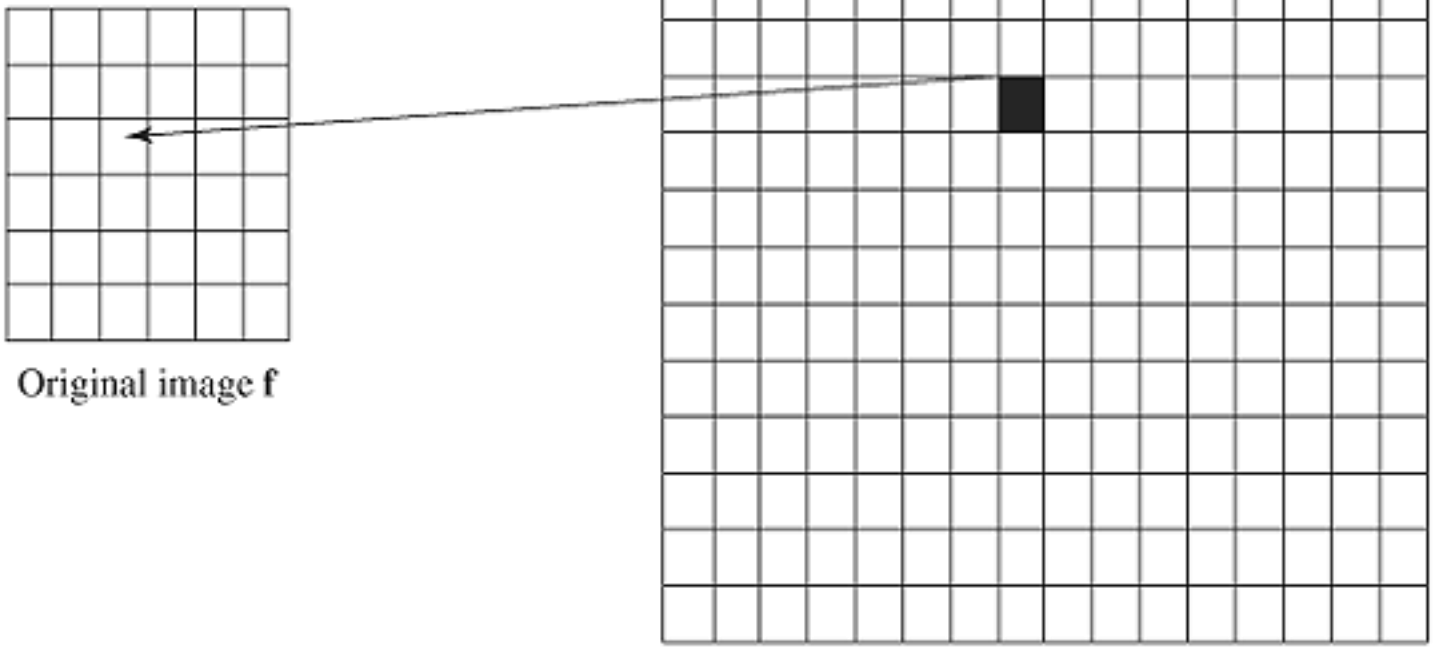
The first two steps in resampling - 1



The first two steps in resampling - 2

Step 2. Map each pixel in the scaled image back to a position in the original image.

Position (6,7) in scaled image maps back to position (2.25, 2.625) in original image.



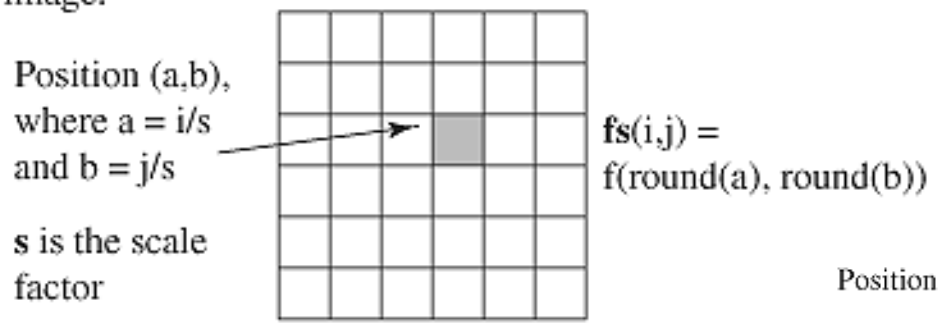
Original image **f**

Scaled image **fs**

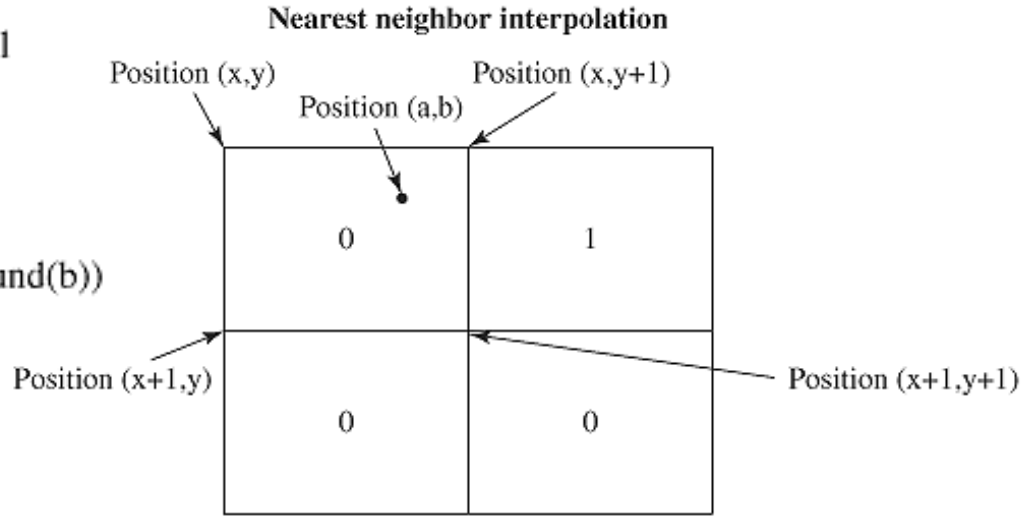
Nearest neighbor interpolation

- ***Nearest neighbor interpolation*** simply rounds down to find one close pixel whose value is used for ***fs***(*i, j*)

The **nearest neighbor algorithm** assigns to **fs**(i,j) the color value f(round(a), round(b)) from the original image.



The nearest neighbor is marked in gray.



Position with coordinates closest to both a and b gets a 1 in the convolution kernel.

Bilinear interpolation

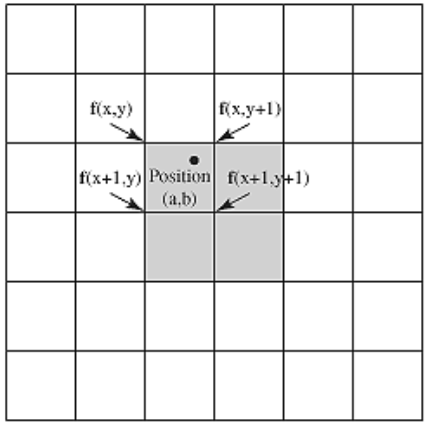
- ***Bilinear interpolation*** uses four neighbors and makes ***fs***(*i, j*) a weighted sum of their color values. The contribution of each pixel toward the color of ***fs***(*i, j*) is a function of how close the pixel's coordinates are to (*a, b*).

Bilinear interpolation uses an average color value of the four pixels surrounding position (a,b) in the original image. Each neighbor's contribution to the color is based on how close it is to (a,b). Let

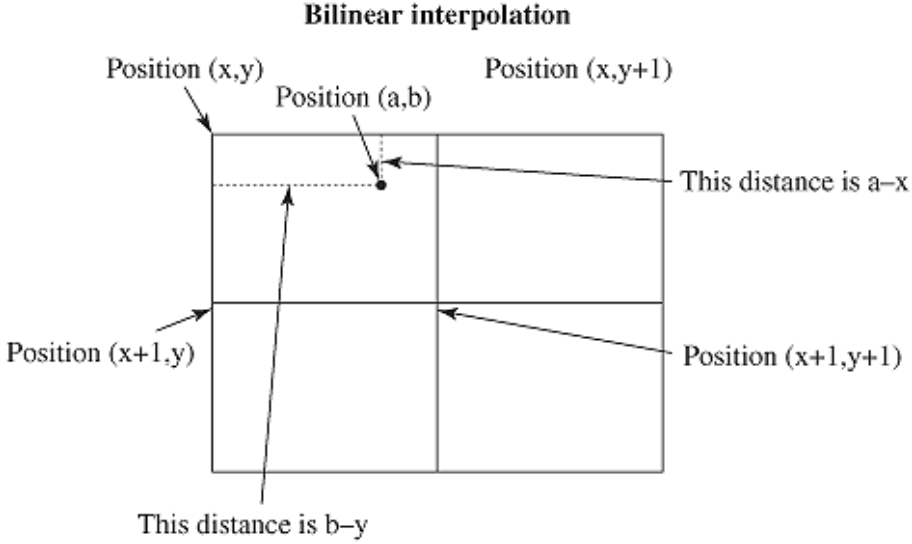
x = floor(a)
y = floor(b)

Then the pixels surrounding position (a,b) are

f(x, y)
f(x+1, y)
f(x+1, y+1)
f(x, y+1)



Neighborhood is shown in gray.



The color of the pixel in image **fs** is a weighted average of the four neighboring pixels. Weights come from each pixel's proximity to (a,b).

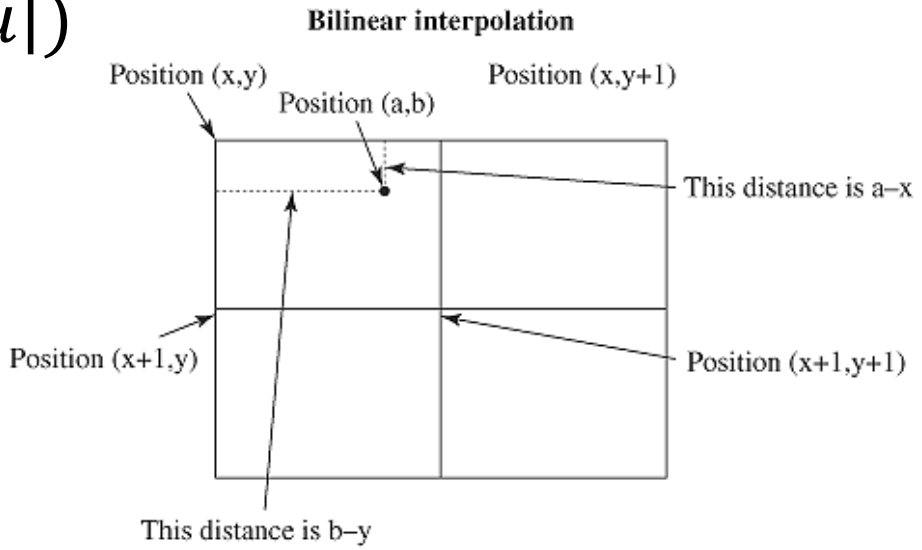
Bilinear interpolation

- 2 x 2 convolution mask H

$$t(m,n) = a - (x+m), 0 \le m \le 1, 0 \le n \le 1$$

$$u(m,n) = b - (y+n), 0 \le m \le 1, 0 \le n \le 1$$

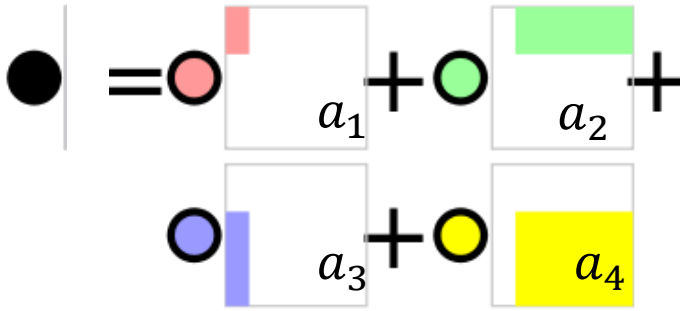
$$H(m,n) = (1-|t|)(1-|u|)$$

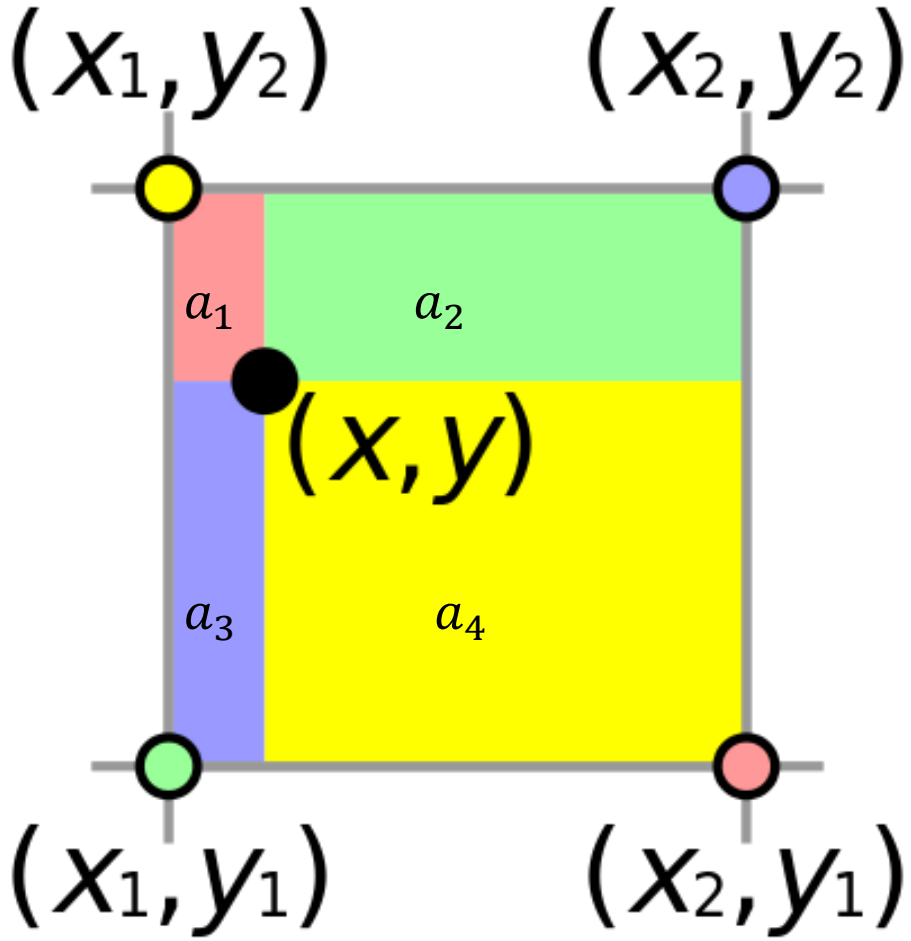


The color of the pixel in image **fs** is a weighted average of the four neighboring pixels. Weights come from each pixel's proximity to (a,b).

Bilinear interpolation

- Value of (x, y) can be determined:

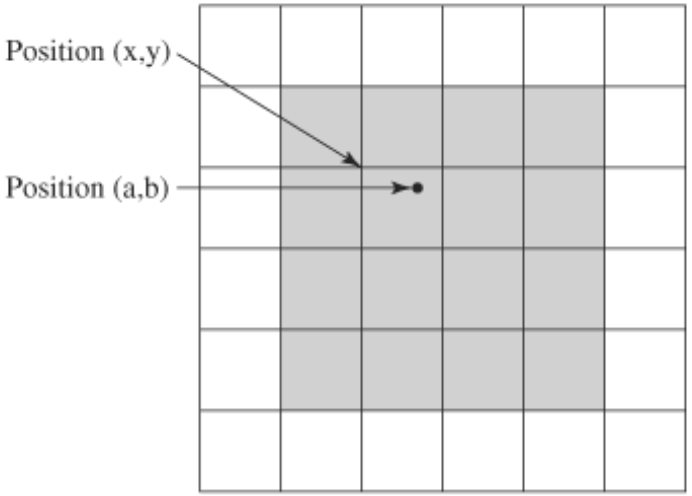




Bicubic interpolation

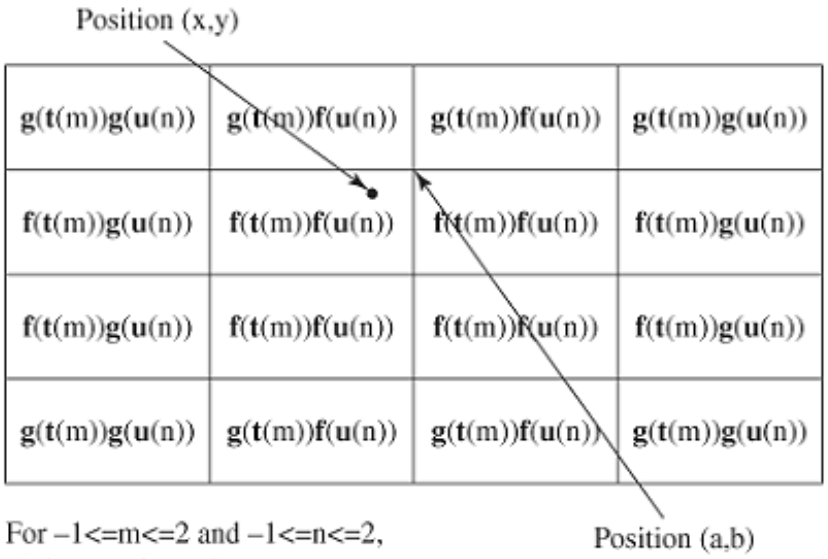
- ***Bicubic interpolation*** uses a neighborhood of sixteen pixels to determine the value of ***fs***(*i, j*)

Bicubic interpolation uses an "average" color value of the 16 pixels surrounding position (a,b) in the original image. The weight of each neighbor's contribution is based on a cubic equation that accounts for how close each neighbor is.



Neighbors are shaded in gray. The neighborhood of (a,b) extends from x–1 to x+2 in the vertical direction and from y–1 to y+2 in the horizontal direction.

Bicubic interpolation



For $-1 \le m \le 2$ and $-1 \le n \le 2$,

$t(m) = a - (x + m)$

$u(n) = b - (y + n)$

$f(t(m)) = 1 - 2t(m)^2 + |t(m)|^3$

$f(u(n)) = 1 - 2u(n)^2 + |u(n)|^3$

$g(t(m)) = 4 - 8|t(m)| + 5t(m)^2 - |t(m)|^3$

$g(u(n)) = 4 - 8|u(n)| + 5u(n)^2 - |u(n)|^3$

4x4 convolution mask

Interpolation Comparison

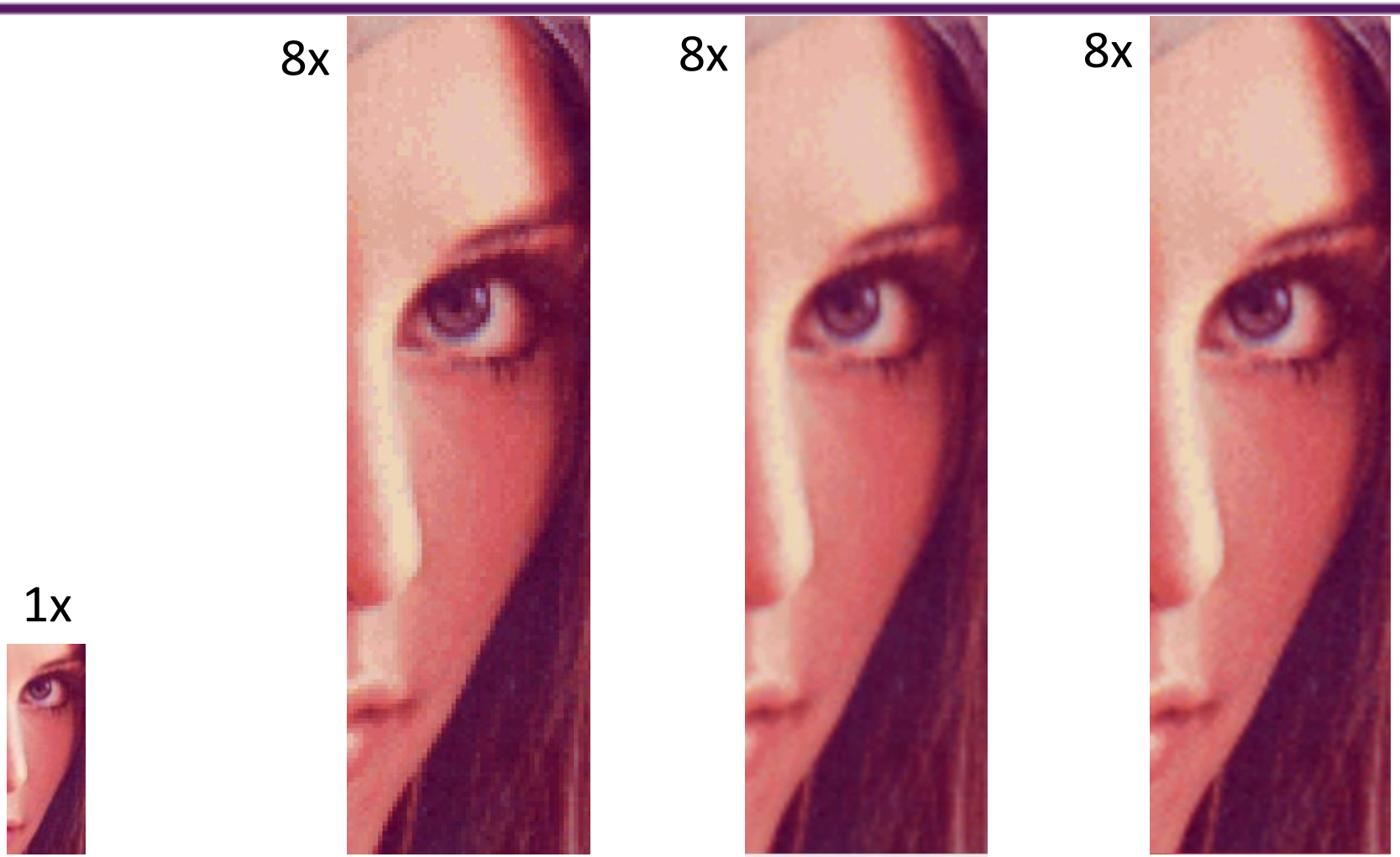


Image compression

- Image compression may be **lossy** or **lossless**.
- Lossless compression is preferred for archival purposes and often for medical imaging, technical drawing or comics.
- Lossy compression is suitable for natural images where minor loss of fidelity is acceptable.

Huffman encoding

- ***Huffman encoding*** is another lossless compression algorithm that is used on bitmap image files.
- A ***variable-length encoding*** scheme; that is, not all color codes use the same number of bits.
- The Huffman encoding algorithm requires two passes:
  1) determining the codes for the colors
  2) compressing the image file by replacing each color with its code

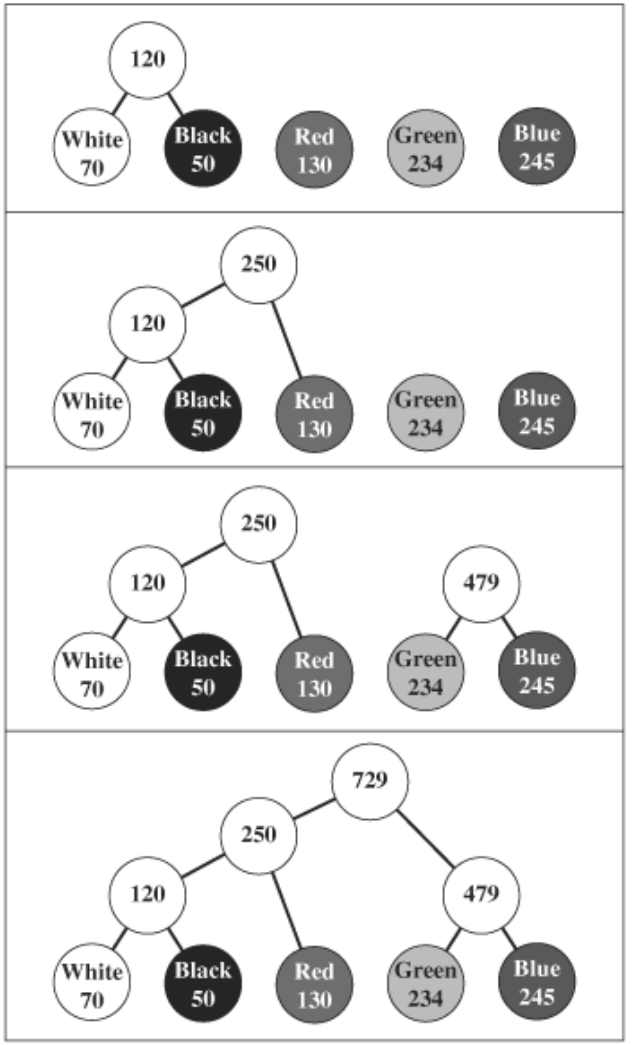
Huffman encoding - example

- The image file has only 729 pixels in it, with the following colors and the corresponding frequencies
  - White 70
  - Black 50
  - Red 130
  - Green 234
  - Blue 245

- A node is created for each of the colors in the image, with the frequency of that color's appearance stored in the node

Huffman encoding - example

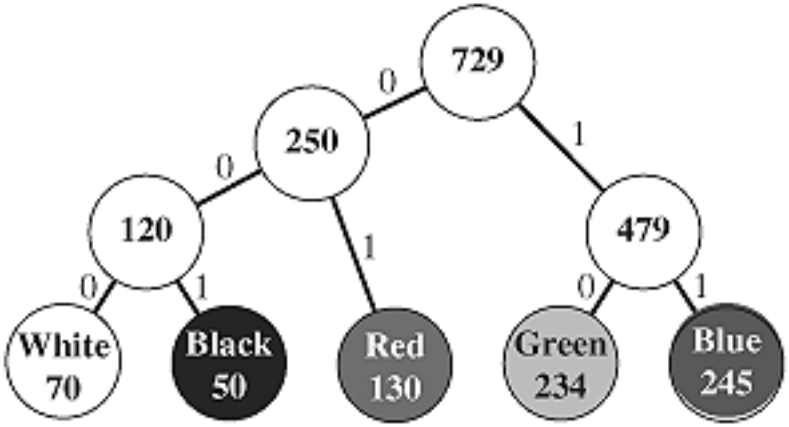
- Now the two nodes with the smallest value for *freq* are joined such that they are the children of a common parent node, and the parent node's *freq* value is set to the sum of the *freq* values in the children nodes
- This node-combining process repeats until you arrive at the creation of a root node



Huffman encoding - example

- Once the tree has been created, the branches are labeled with 0s on the left and 1s on the right
- After the codes have been created, the image file can be compressed using these codes.

Label branches with 0s on the left and 1s on the right.



For each leaf node, traverse tree from root to leaf node and gather code for the color associated with the leaf node.

White	000
Black	001
Red	01
Green	10
Blue	11

Note that not all codes are the same number of bits, and no code is a prefix of any other code.

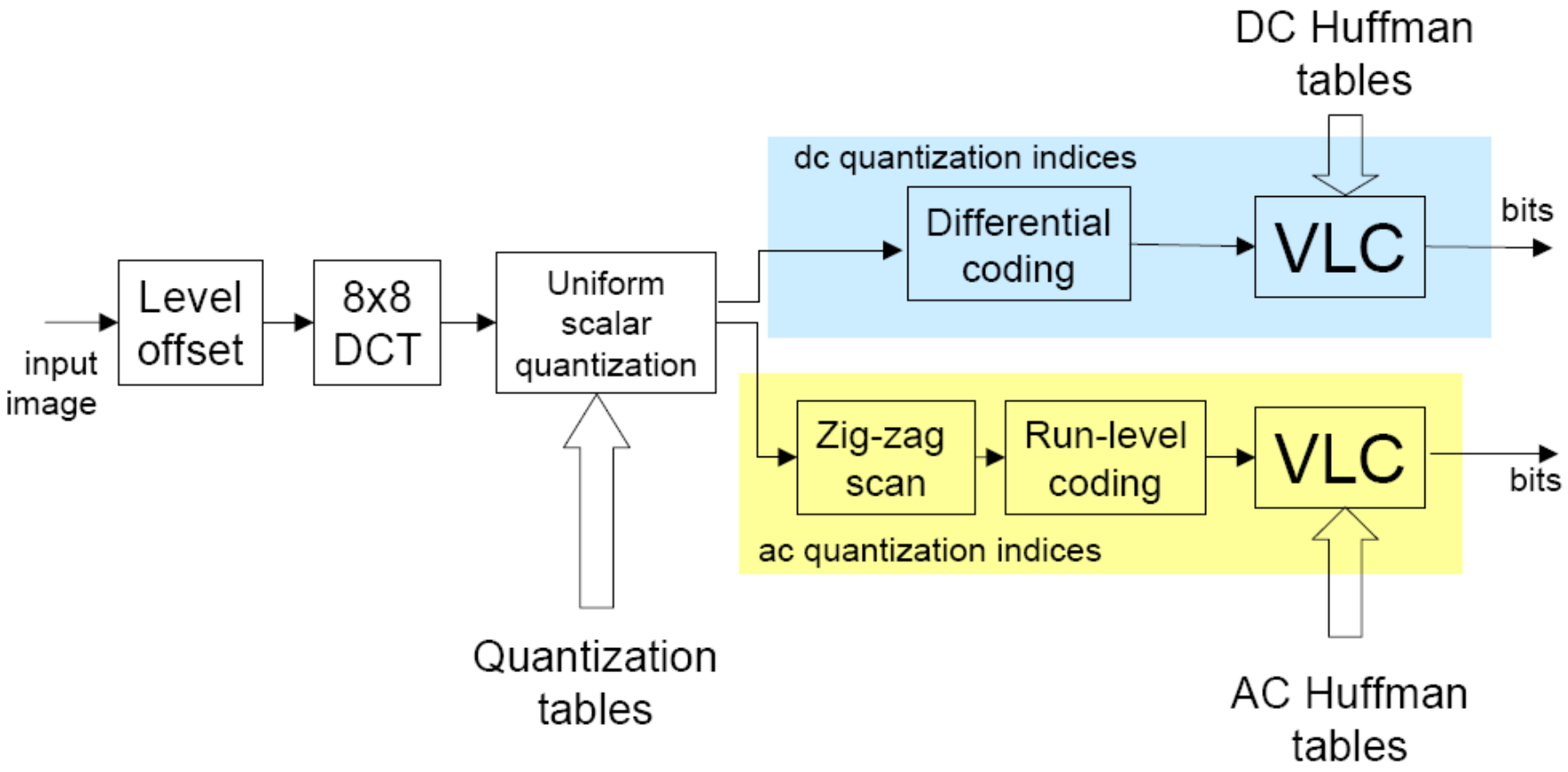
Huffman encoding

- By combining least-valued nodes first and creating the tree from the bottom up, the algorithm ensures that the colors that appear least frequently in the image have the longest codes
- Also, because the codes are created from the tree data structure, no code can be a prefix of another code
- Huffman encoding is useful as a step in JPEG compression, as we will see in the next section.

JPEG Compression

- JPEG is a lossy compression method
- Image processing programs allow you to choose the JPEG compression rate
- The main disadvantage to JPEG compression is that it takes longer for the encoding and decoding than other algorithms require

JPEG Image Compression



JPEG Compression – step1

- **Step1 : Divide the image into 8 × 8 pixel blocks and convert RGB to a luminance/chrominance color model**
- The image is divided into 8 × 8 pixel blocks to make it computationally more manageable for the next steps. Converting color to a luminance/chrominance model makes it possible to remove some of the chrominance information, to which the human eye is less sensitive, without significant loss of quality in the image.

RGB → YC_bC_r

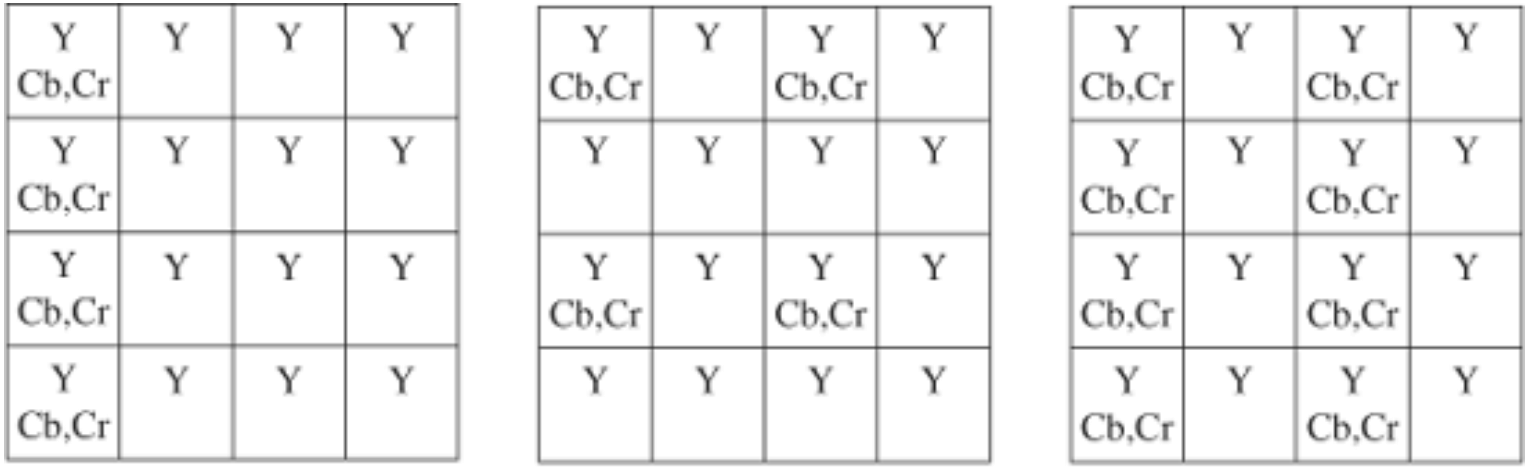
- YC_bC_r color model represents color in terms of one luminance component, Y, and two chrominance components, C_b and C_r.
- The human eye is more sensitive to changes in light (*i.e.*, luminance) than in color (*i.e.*, chrominance).

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

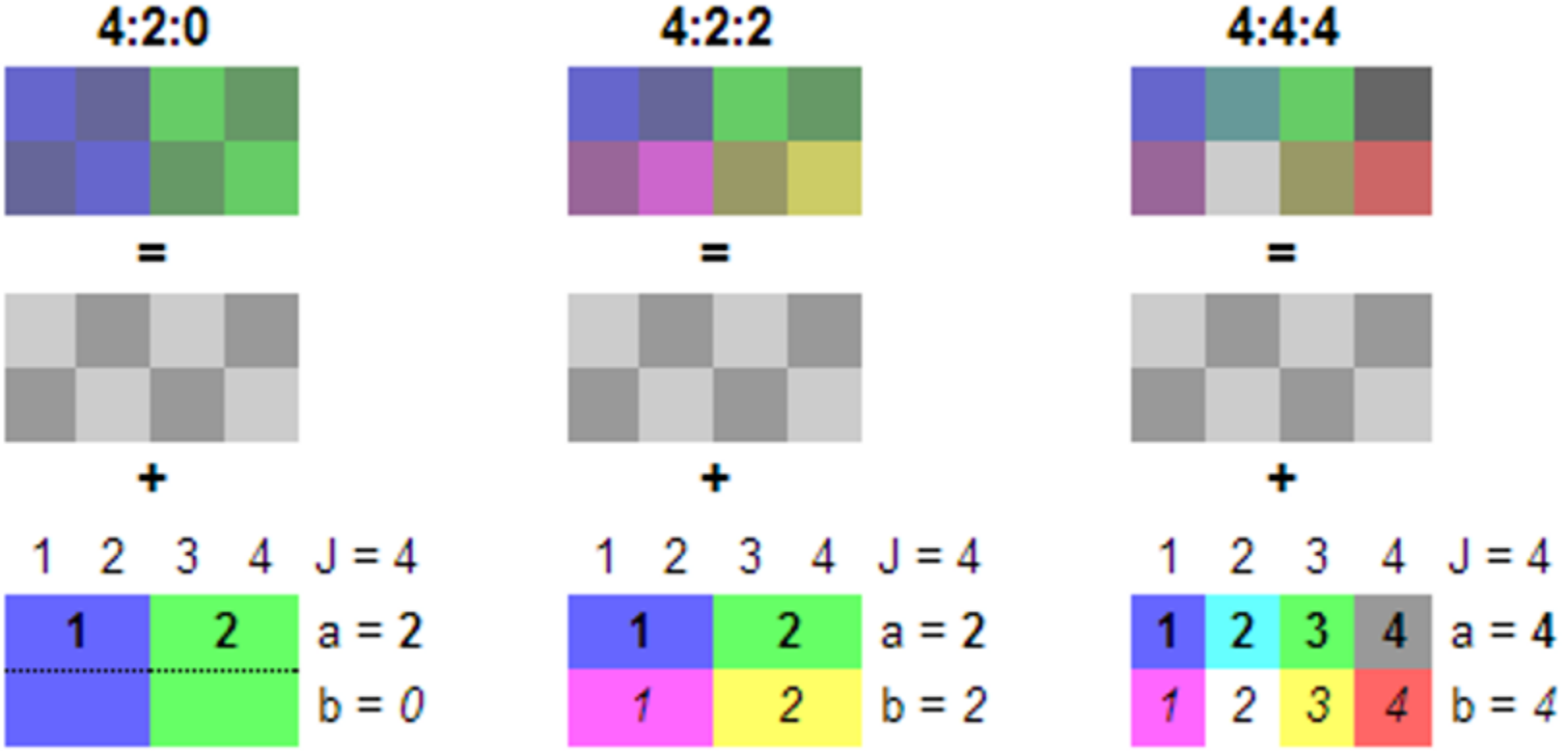
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.164 & 0 & 1.596 \\ 1.164 & -0.392 & 0.439 \\ 1.164 & 2.017 & 0 \end{bmatrix} \left[\begin{bmatrix} Y - 16 \\ C_b - 128 \\ C_r - 128 \end{bmatrix} \right]$$

Chrominance subsampling

- luminance/chrominance subsampling is represented in the form *a*:*b*:*c*
- For each pair of four-pixel-wide rows, *a* is the number of Y samples in both rows, *b* and c are the numbers of C_b (C_r) samples in the 1st and 2nd rows, respectively.



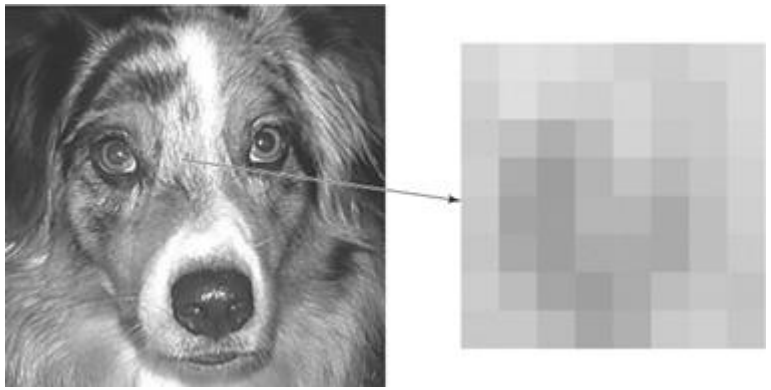
Chrominance subsampling



JPEG Compression – step2

- **Step 2: Shift values by –128 and transform from the spatial to the frequency domain**
- On an intuitive level, shifting the values by –128 is like looking at the image function as a waveform that cycles through positive and negative values.
- This step is a preparation for representing the function in terms of its frequency components. Transforming from the spatial to the frequency domain makes it possible to remove high frequency components.
- High frequency components are present if color values go up and down quickly in a small space. These small changes are barely perceptible in most people's vision, so removing them does not compromise image quality significantly.

JPEG Compression – step2



Grayscale Values for 8 × 8 Pixel Area							
222	231	229	224	216	213	220	224
216	229	217	215	221	210	209	223
211	202	283	198	218	207	209	221
214	180	164	188	203	193	205	217
209	171	166	190	190	178	199	215
206	177	166	179	180	178	199	210
212	197	173	166	179	198	206	203
208	208	195	174	184	210	214	206

→

Pixel Values for Image in Figure 3.49 Shifted by –128							
94	103	101	96	88	85	92	96
88	101	89	87	93	82	81	95
83	74	55	70	90	79	81	93
86	52	36	60	75	65	77	89
81	43	38	62	62	50	71	87
78	49	38	51	52	50	71	82
84	69	45	38	51	70	78	75
80	80	67	46	56	82	86	78

JPEG Compression – step2

- Take the two-dimensional DCT step.

DCT of an 8 × 8 Pixel Area							
585.7500	–24.5397	59.5959	21.0853	25.7500	–2.2393	–8.9907	1.8239
78.1982	12.4534	–32.6034	–19.4953	10.7193	–10.5910	–5.1086	–0.5523
57.1373	24.829	–7.5355	–13.3367	–45.0612	–10.0027	4.9142	–2.4993
–11.8655	6.9798	3.8993	–14.4061	8.5967	12.9151	–0.3122	–0.1844
5.2500	–1.7212	–1.0824	–3.2106	1.2500	9.3595	2.6131	1.1199
–5.9658	–4.0865	7.6451	13.0616	–1.1927	1.1782	–1.0733	–0.5631
–1.2074	–5.7729	–2.0858	–1.9347	1.6173	2.6671	–0.4645	0.6144
0.6362	–1.4059	–0.719	1.6339	–0.1438	0.2755	–0.0268	–0.2255

8 × 8 DCT Bases

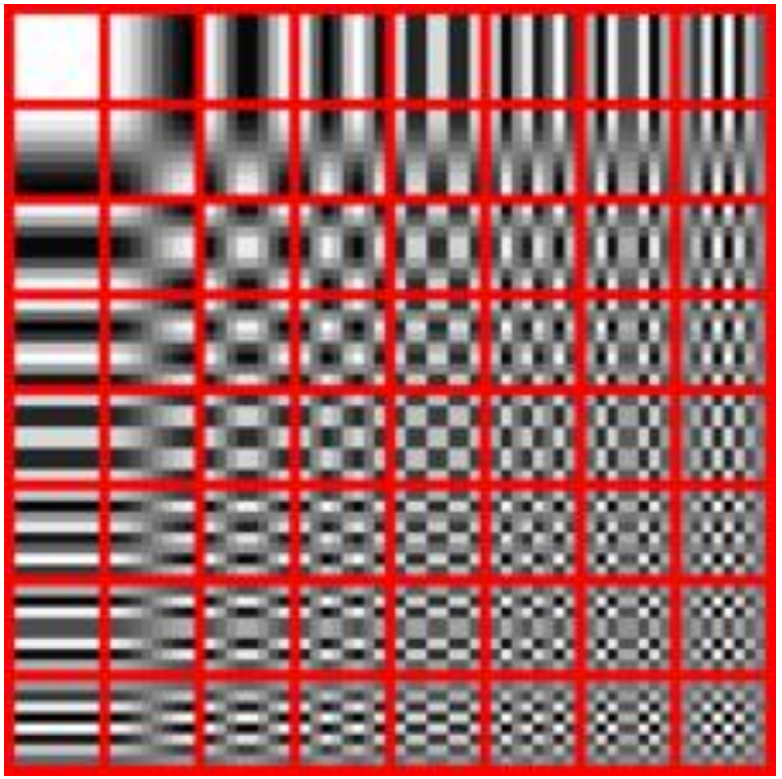
$$G_{u,v} = \sum_{x=0}^{7}\sum_{y=0}^{7} \alpha(u)\alpha(v) g_{x,y} \cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

DCT coefficients

Image

→ *u*

↓ *v*



Inverse DCT

$$f_{x,y} = \sum_{u=0}^{7}\sum_{v=0}^{7} \alpha(u)\alpha(v) F_{u,v} \cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

JPEG Compression – step3

- **Step 3: Quantize the frequency values**
- Quantization involves dividing each frequency coefficient by an integer and rounding off. The coefficients for high-frequency components are typically small, so they often round down to 0—which means, in effect, that they are thrown away.

$$B_{j,k} = \text{round}\left(\frac{G_{j,k}}{Q_{j,k}}\right) \text{ for } j = 0, 1, 2, \ldots, 7; k = 0, 1, 2, \ldots, 7$$

G is the unquantized DCT coefficients ; Q is the quantization matrix and B is the quantized DCT coefficients

JPEG Compression – step3

Quantization Table for example							
8	6	6	7	6	5	8	7
7	7	9	9	8	10	12	20
13	12	11	11	12	25	18	19
15	20	29	26	31	30	29	26
28	28	32	36	46	39	32	34
44	35	28	28	40	55	41	44
48	49	52	52	52	31	39	57
61	56	50	60	46	51	52	50

Quantized DCT Values							
73	–4	10	3	4	0	–1	0
11	2	–4	–2	1	–1	0	0
4	2	–1	–1	–4	0	0	0
–1	0	0	–1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Standard JPEG Quantization Tables

- Luminance

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	36	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

- Chrominance

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Quantization Tables - Quality Factor

- Quantization tables can be scaled to a **quality factor Q_f**. The quality factor allows the image creation device to choose between larger, higher quality images and smaller, lower quality images.
- The value of Q_f can range between 1 and 100 and is used to compute the **scaling factor, S**.

$$S = (Q_f < 50)? \frac{5000}{Q_f} : 200\text{-}2Q_f$$

Quantization Tables - Quality Factor

- Each element i in the scaled table T_s is computed using the i^{th} element in the base table T_b.

$$T_s[i] = \left\lfloor \frac{S * T_b[i] + 50}{100} \right\rfloor$$

Ex:

$Q_f = 80 \Rightarrow S = 40$

$6 = \lfloor (40 * 16 + 50)/100 \rfloor$

- Luminance

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Standard JPEG quantization table

6	4	4	6	10	16	20	24
5	5	6	8	10	23	24	22
6	5	6	10	16	23	28	22
6	7	9	12	20	35	32	25
7	9	15	22	27	44	41	31
10	14	22	26	32	42	45	37
20	26	31	35	41	48	48	40
29	37	38	39	45	40	41	40

Standard JPEG quantization table scaled with **Q_f = 80**

Difference of Quality factor

Quality = 100

Quality = 10

Quality = 50

Quality = 1

JPEG Compression – step4

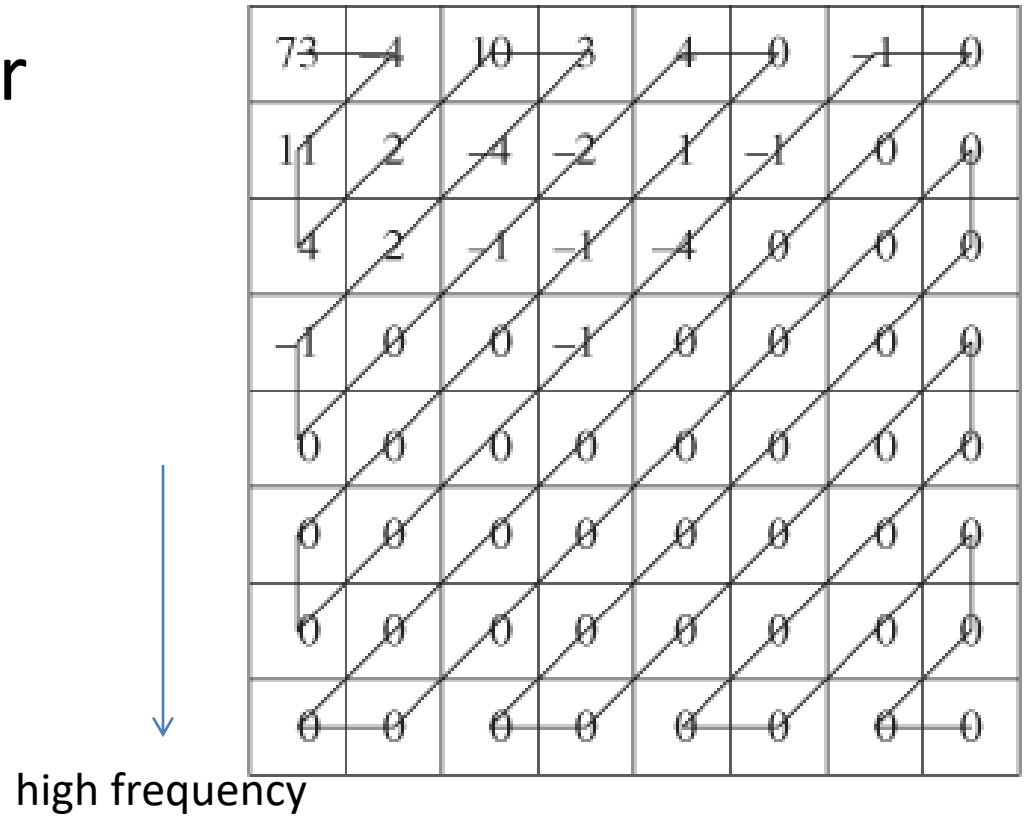
- **Step 4: Apply DPCM to the block**
- ***DPCM*** is the abbreviation for ***differential pulse code modulation***. In this context, DCPM is simply storing the difference between the first value in the previous 8×8 block and the first value in the current block. Since the difference is generally smaller than the actual value, this step adds to the compression.

JPEG Compression – step5

- **Step 5: Arrange the values in a zigzag order and do run-length encoding.**
- The zigzag reordering sorts the values from low-frequency to high-frequency components. The high-frequency coefficients are grouped together at the end. If many of them round to zero after quantization, run-length encoding is even more effective.

JPEG Compression – step5

- Quantized DCT values rearranged from low- to high-frequency components
- Zigzag order



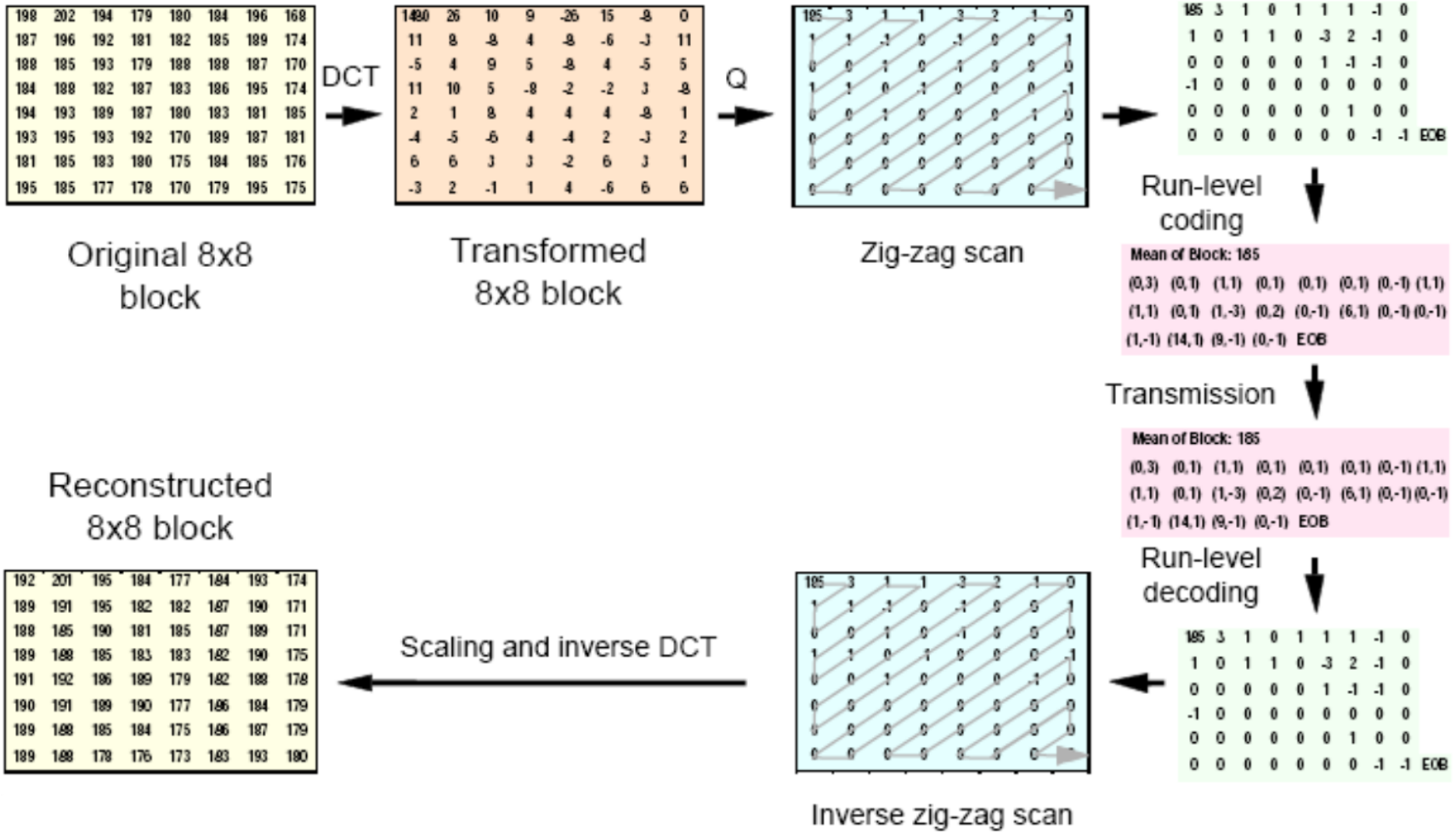
JPEG Compression – step6

- **Step 6: Do entropy encoding.**
- Additional compression can be achieved with some kind of entropy encoding.
- In JPEG, Huffman coding is used.

JPEG Compression Algorithm

```
algorithm jpeg
/*Input: A bitmap image in RGB mode.
Output: The same image, compressed.*/
{
  Divide image into 8 × 8 pixel blocks
  Convert image to a luminance/chrominance model such as YCbCr (optional)
  Shift pixel values by subtracting 128
  Use discrete cosine transform to transform the pixel data from the spatial domain
  to the frequency domain
  Quantize frequency values
  Store DC value (upper left corner) as the difference between current DC value and
  DC from previous block
  Arrange the block in a zigzag order
  Do run-length encoding
  Do entropy encoding (e.g., Huffman)
}
```

Transform Coding



Example of JPEG Compression-GIMP

1.Quality factor
2.Chroma subsampling
3.DCT

Summary

- Bitmapped images
- Aliasing & Anti-aliasing
- Color models & Color quantization
- Dithering
- Image transform
- Resampling
  - interpolation
- Image Compression
  - Hoffman encoding , JPEG