Image Formation and Processing

COMPUTER VISION (HY24011)

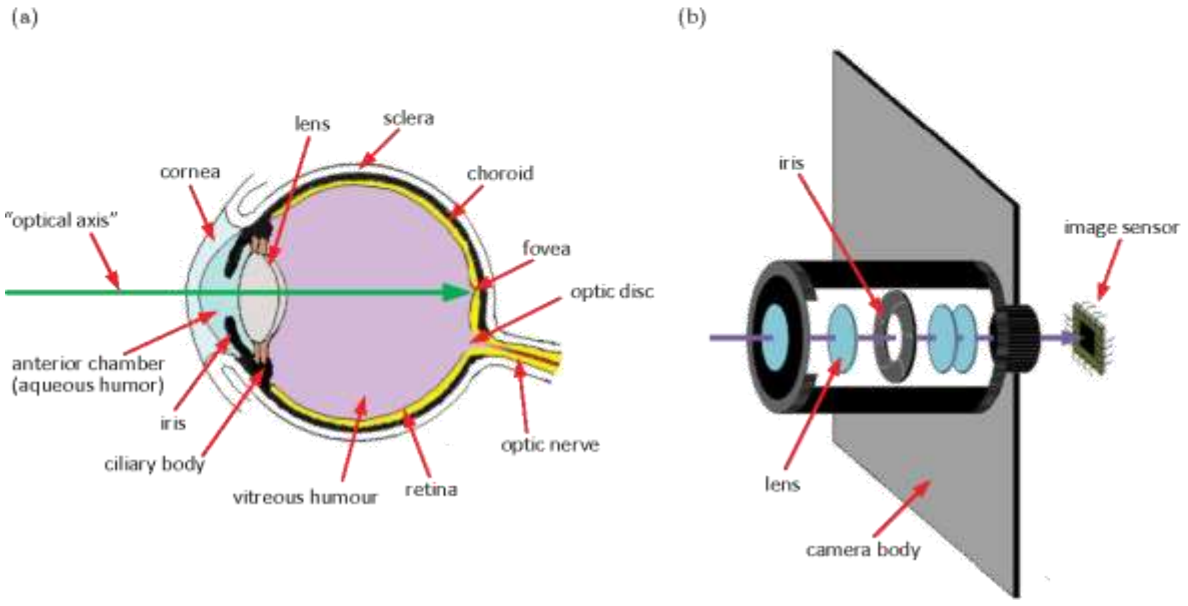
HANYANG UNIV. ERICA

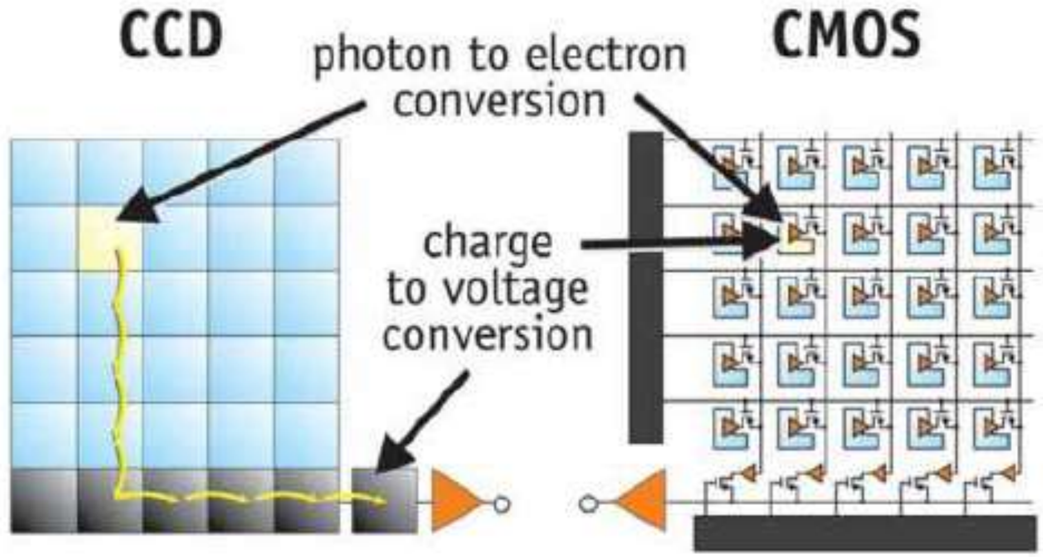
Q YOUN HONG (홍규연)

Contents

- **Image**
- Color models
- Image processing operators
  - Point operators
  - Area-based operators
  - Geometric transformations

Digital Camera vs. Human Eye



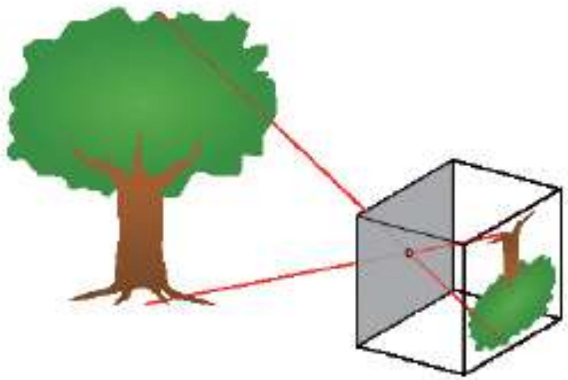


- Similarities between human eye and digital camera
  - Lens, iris(aperture), …
  - Retina corresponds to image sensors (i.e. CCD, CMOS)

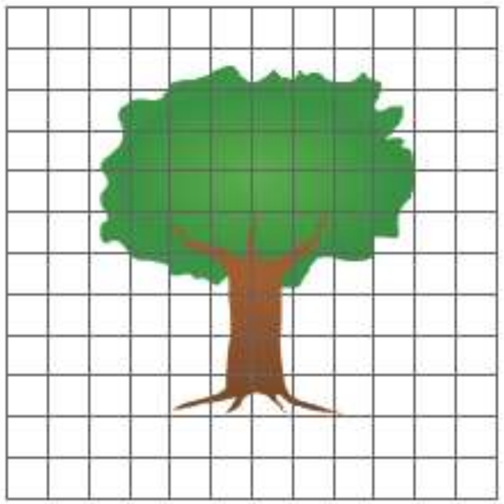
- Digital imaging sensors
  - CCD (Charge-coupled device): move photogenerated charge from pixel to pixel and convert it to the voltage at output node
  - CMOS (Complementary metal oxide on silicon): convert charge to the voltage at each pixel

Digital Image

- Sampling and quantization
  - 2D image space is sampled by M x N (M x N: resolution)
  - Brightness (light intensity) is quantized as L levels ($L \in [0, L-1]$)



Pinhole camera model

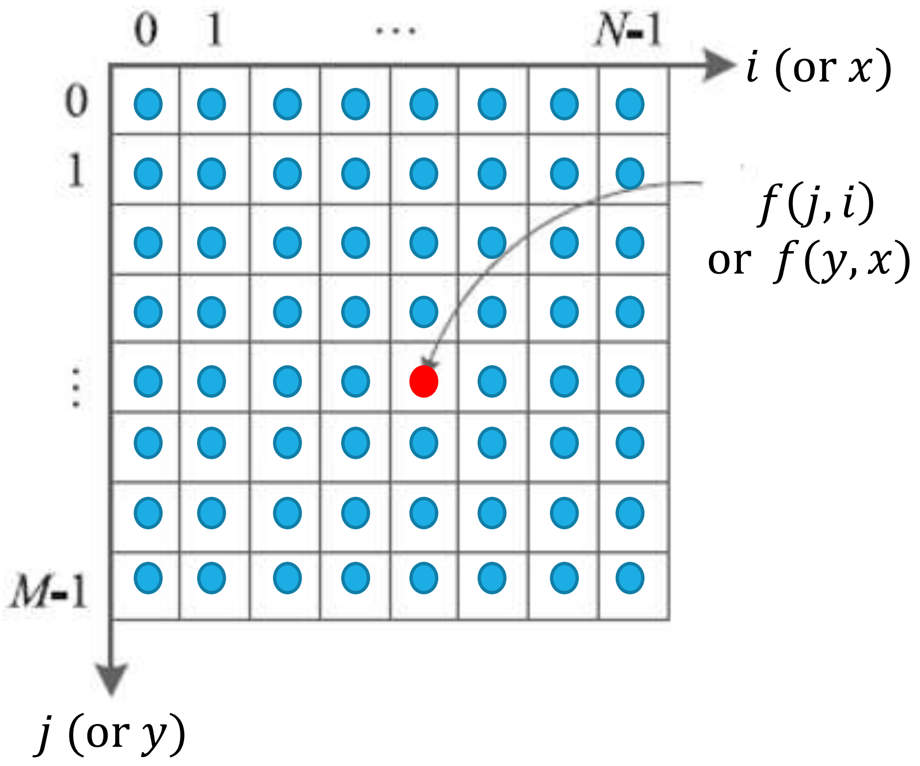


Sampling (M=N=12)

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	3	4	2	3	4	3	0	0	0
0	0	3	7	8	8	8	7	6	3	0	0
0	0	4	8	9	9	9	8	7	5	1	0
0	0	4	7	8	9	9	8	7	5	0	0
0	0	3	6	7	8	8	7	7	3	0	0
0	0	0	2	4	7	8	4	3	0	0	0
0	0	0	0	0	4	7	0	0	0	0	0
0	0	0	0	0	5	6	0	0	0	0	0
0	0	0	0	2	3	4	2	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

Quantization (L=10)

Image Coordinate System



- Image is a 2D array of pixels
- Each pixel location: $f(j,i)$ or $f(y,x)$
  - y, x are integer row, column indices
- Image: $f(\boldsymbol{x})$ or
  $f(j,i), 0 \le j \le M-1, 0 \le i \le N-1$
- Color image: each pixel has three values

$$f_r(\boldsymbol{x}), f_g(\boldsymbol{x}), f_b(\boldsymbol{x})$$

Image Coordinate System

- Sometimes continuous coordinate systems are used
  - $f(y, x)$ is a continuous function where y and x are floats

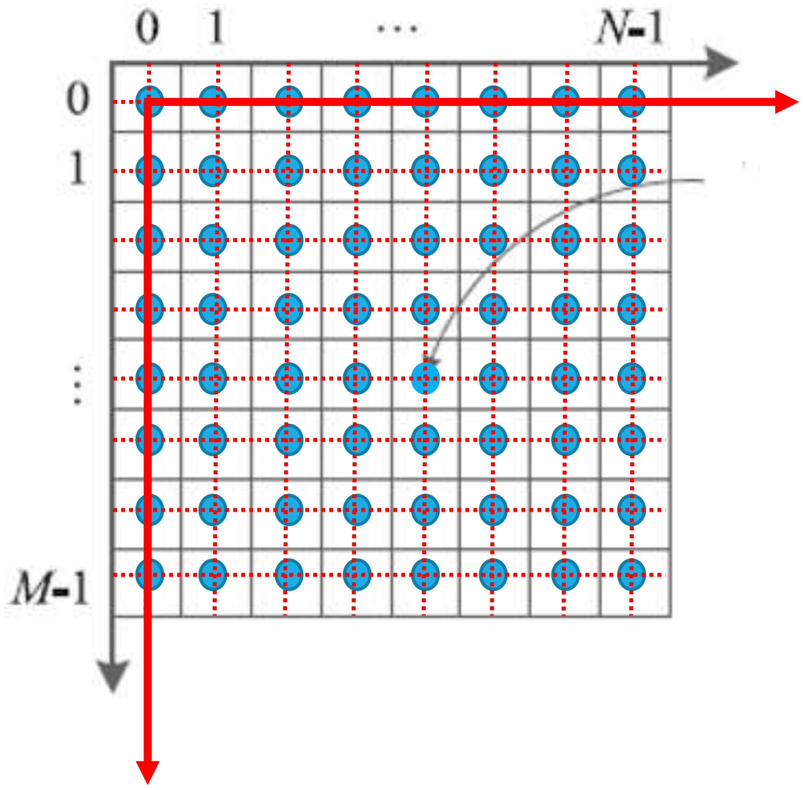
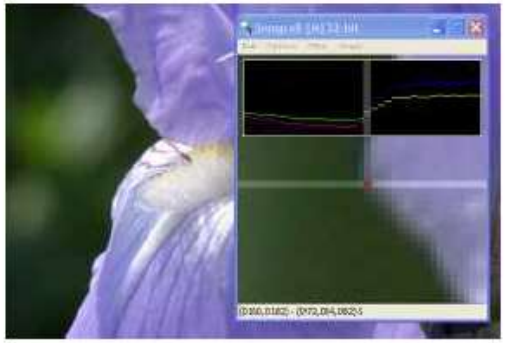


Image Representation

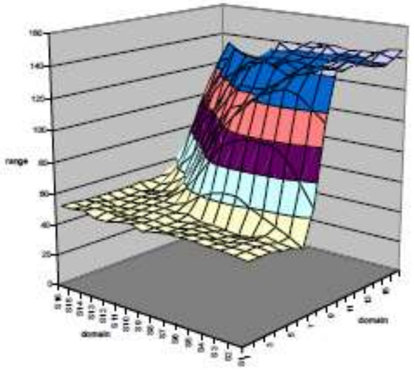
(a) Original Image



(b) Cropped portion
And scanline plot

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

(c) Numeric values
(of grayscaled image)



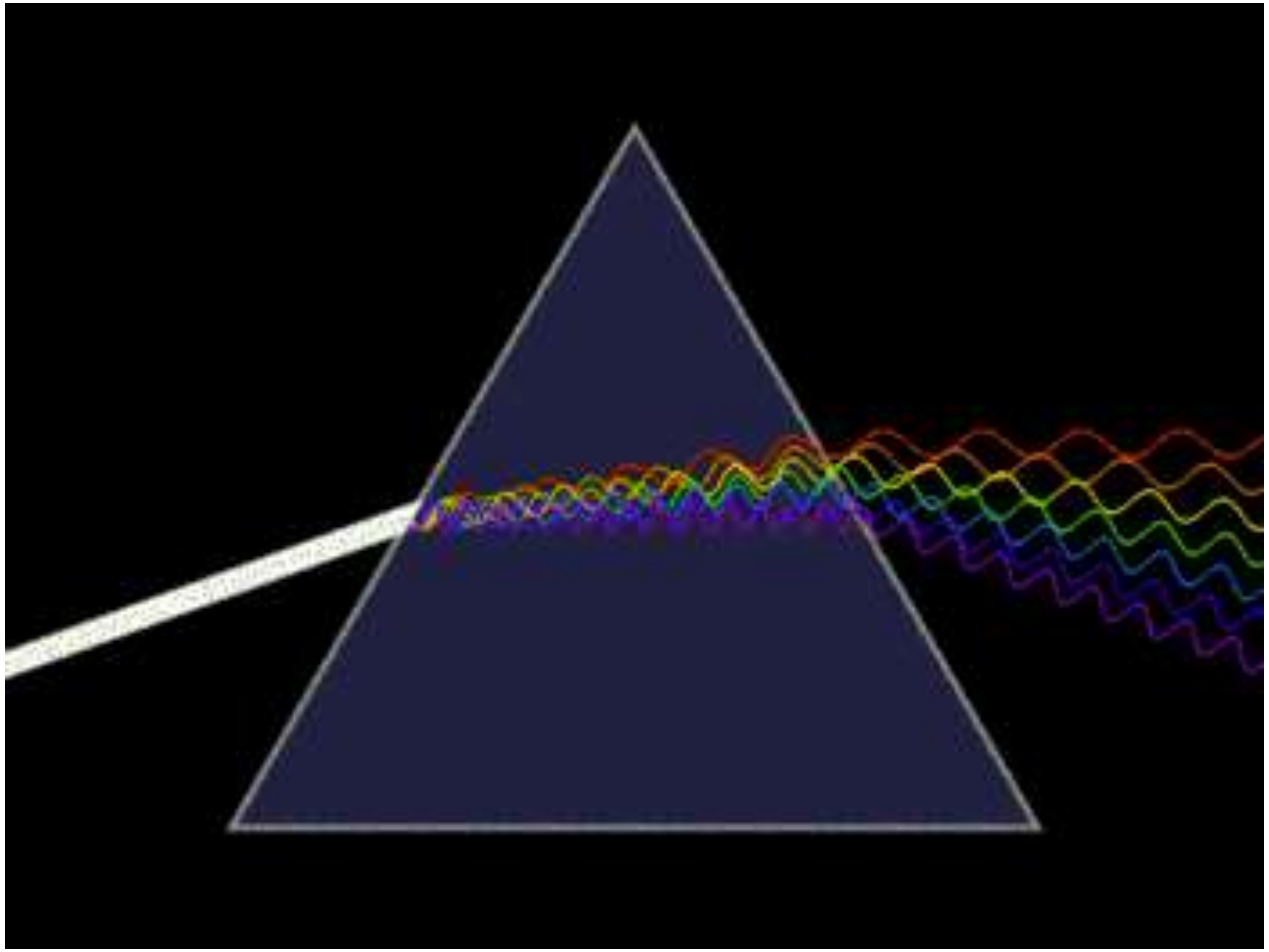
(d) Surface plot
(of grayscaled image)

Contents

- Image
- **Color models**
- Image processing operators
  - Point operators
  - Area-based operators
  - Geometric transformations

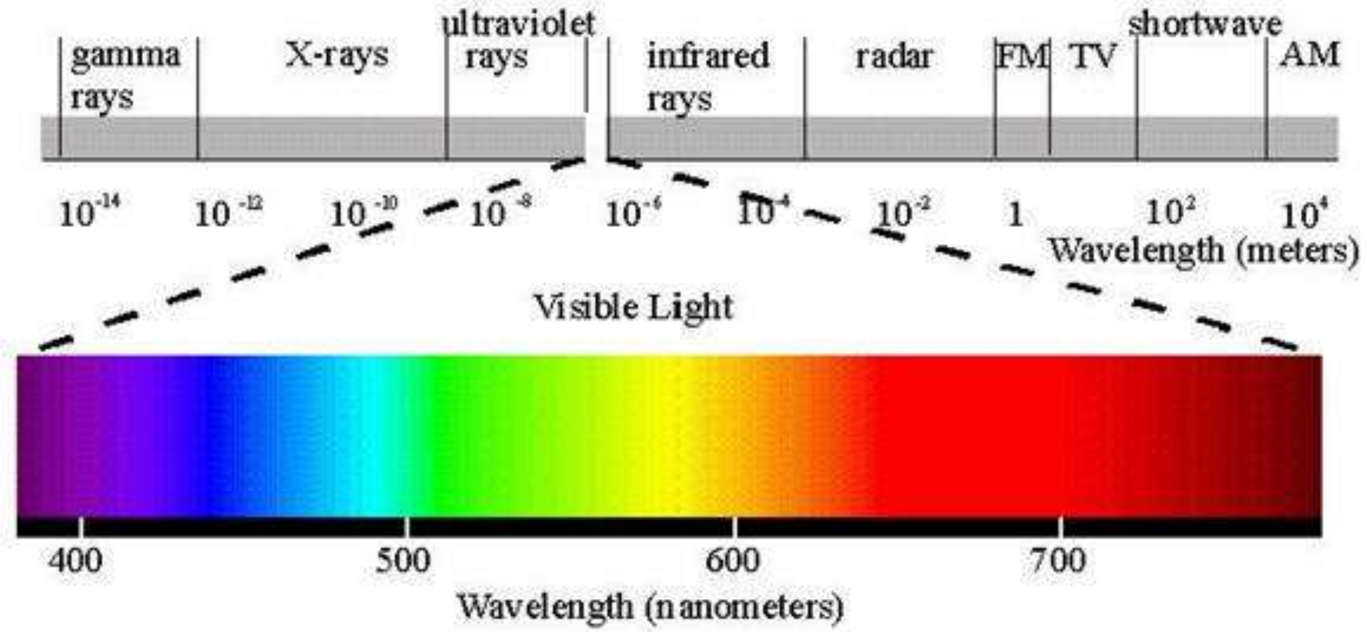
Properties of Light

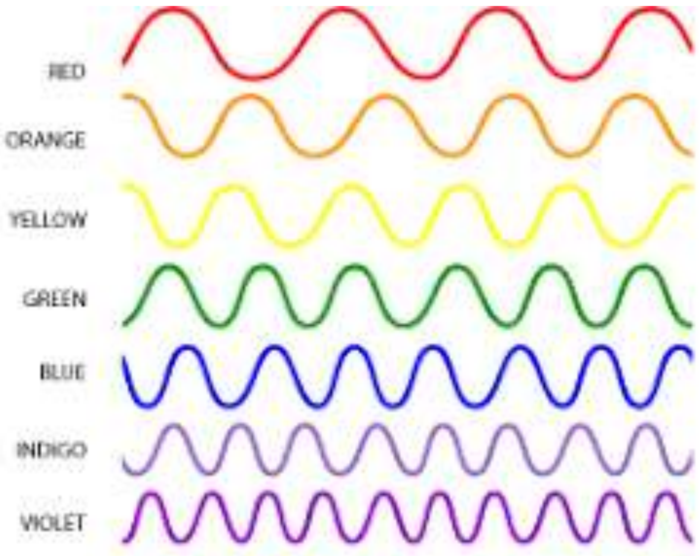
- Light: electromagnetic wave



Properties of Light

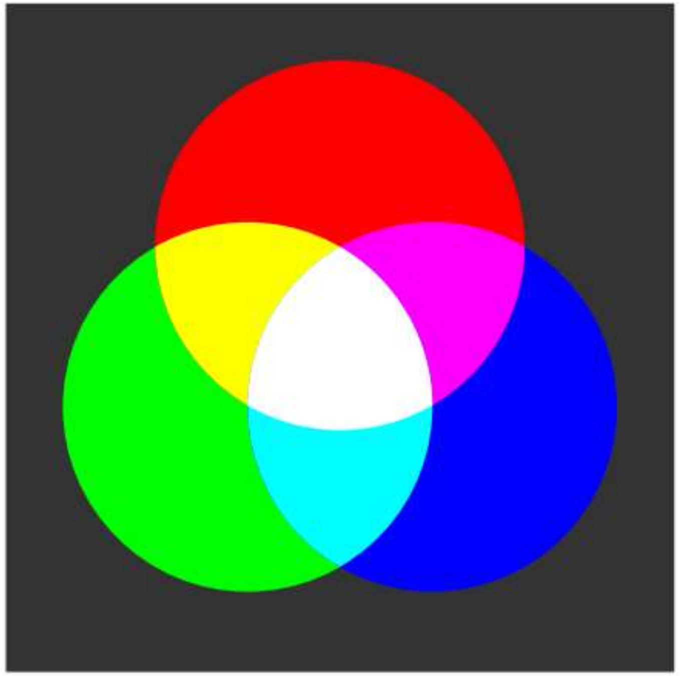
- We only see a small range of wavelengths of light

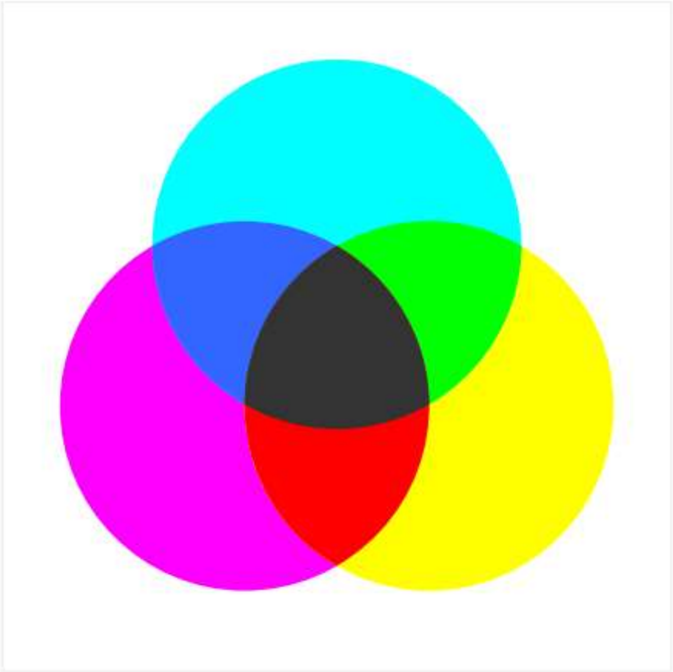




- Short wavelength (380-450nm, 6.7-7.9x10^{14}Hz): violet
- Long wavelength (620-750nm, 4.0-4.8x10^{14}Hz): red

Colors in Painting

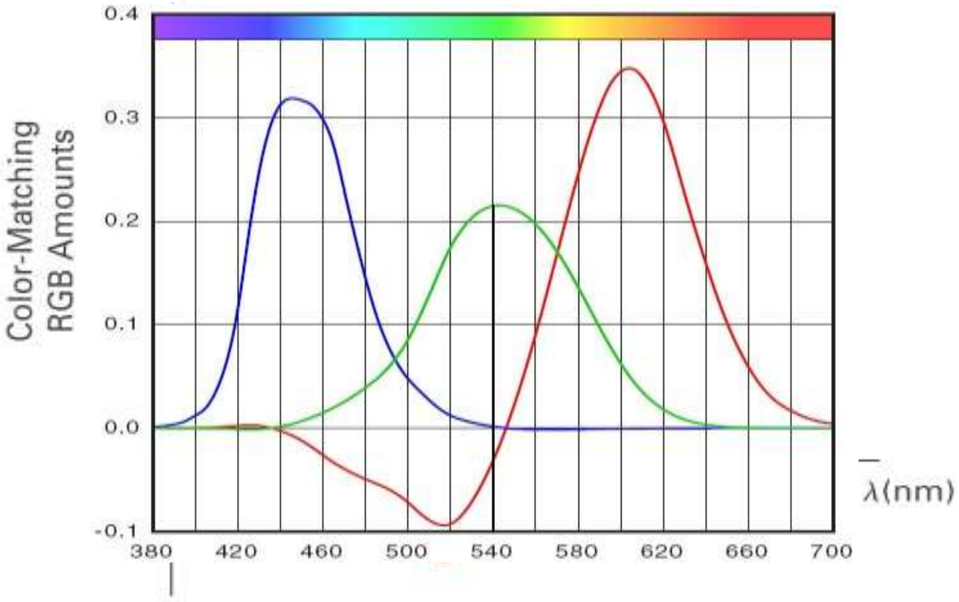


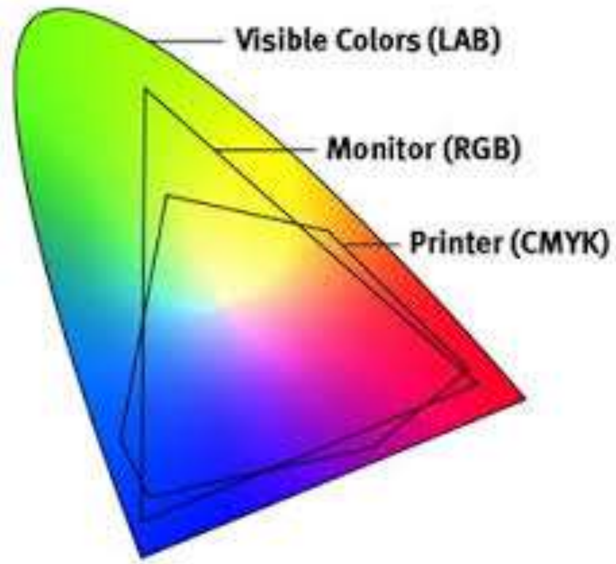


- Additive primary colors red, green, blue mixed to produce cyan, magenta, yellow (left)
- Subtractive primary cyan, magenta, yellow mixed to produce red, green, blue (right)
- Question) Can we mix wavelength to make new color?

Tri-stimulus (tri-chromatic) Theory

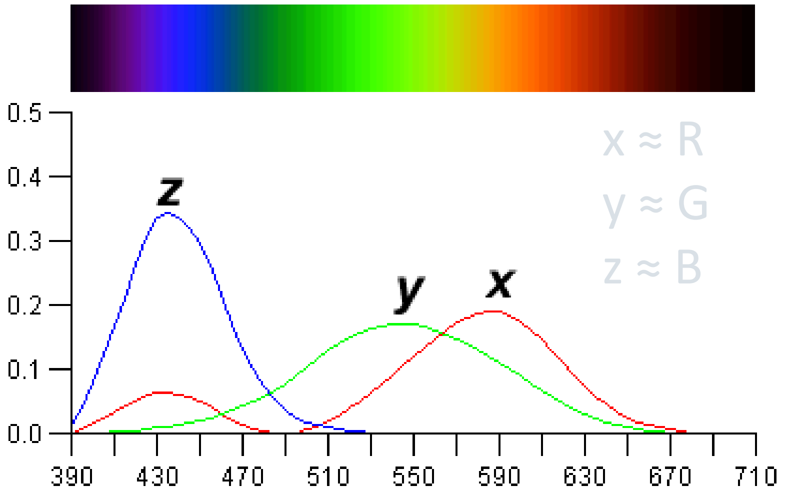
- Tri-stimulus theory: human has three different kinds of cells (cones): each cone responds to different wavelength
- All monochromatic (single wavelength) colors as a mixture of three chosen primaries
- CIE(Commission Internationale d'Eclairage) standardized RGB representation by color matching experiments using primary colors of red(700nm), green(546.1nm), blue(435.8nm)



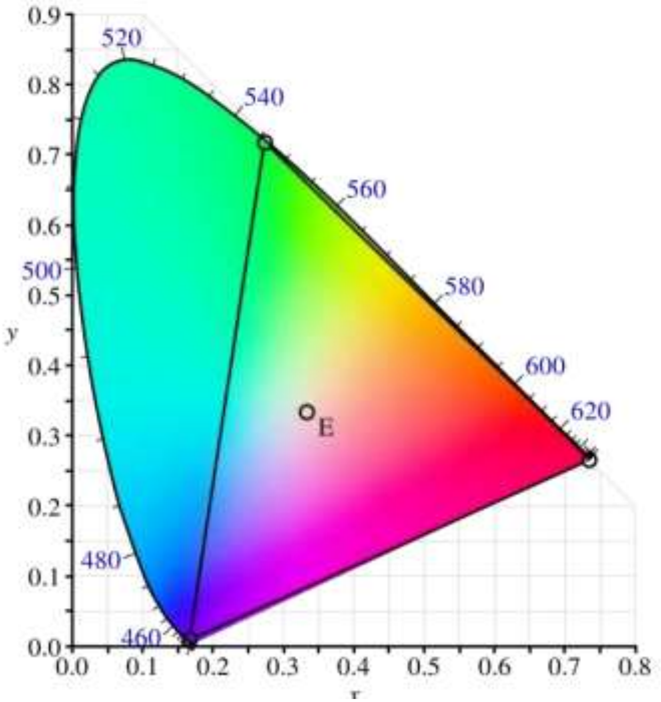


XYZ Color Model

- Color space developed by CIE
  - All spectral colors are defined by positive combination of primary colors X,Y,Z
  - Maps Y-axis to the luminance(perceived relative brightness)



x ≈ R
y ≈ G
z ≈ B



$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

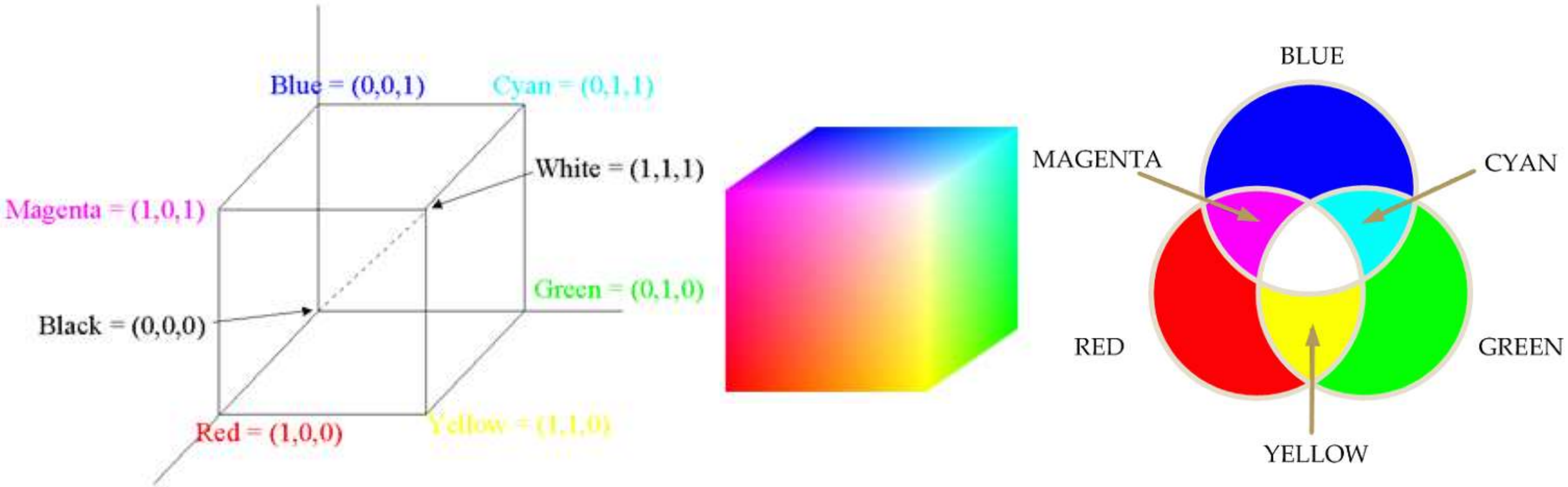
$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z},$$

Chromaticity coordinates:

- X,Y,Z sums up to 1
- Represent pure color without the absolute intensity of the color

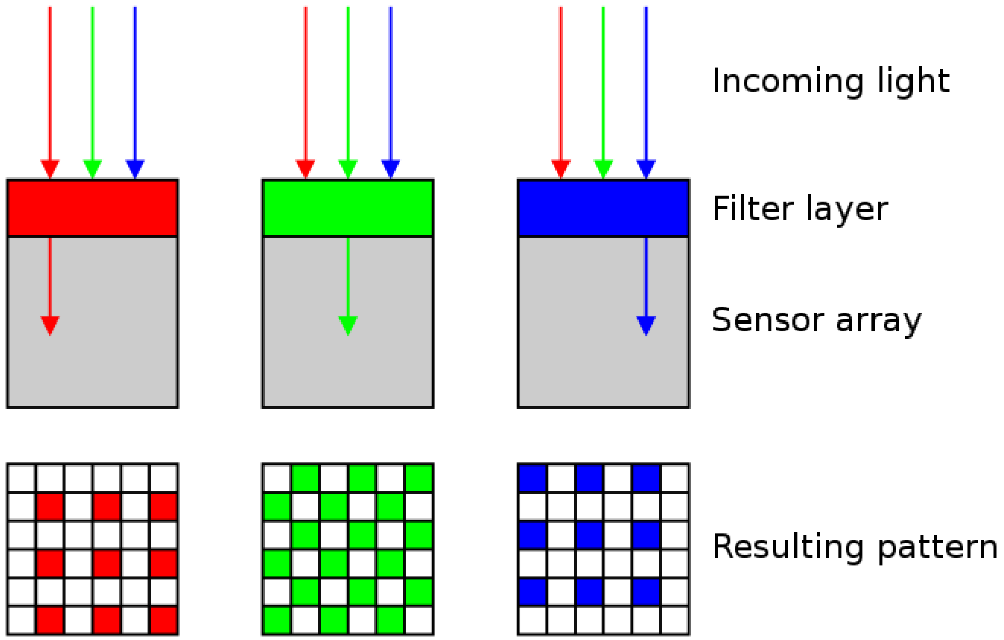
RGB Color Model

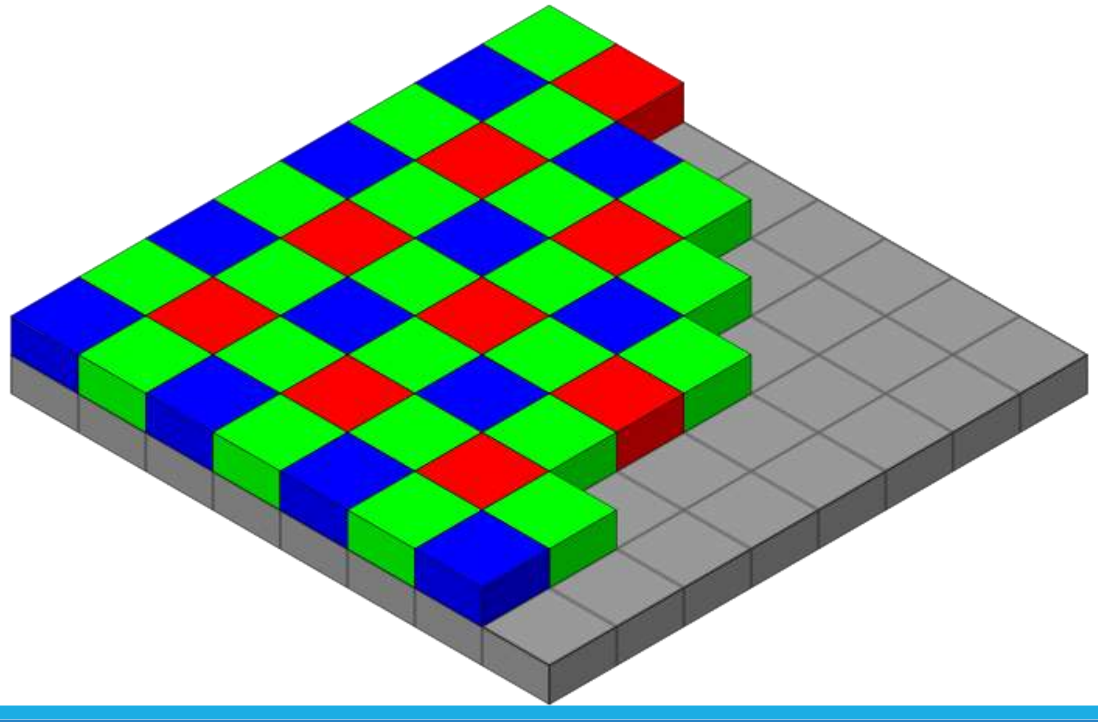
- Based on tri-stimulus theory
- Colors are represented in a unit cube



Color Filter Arrays

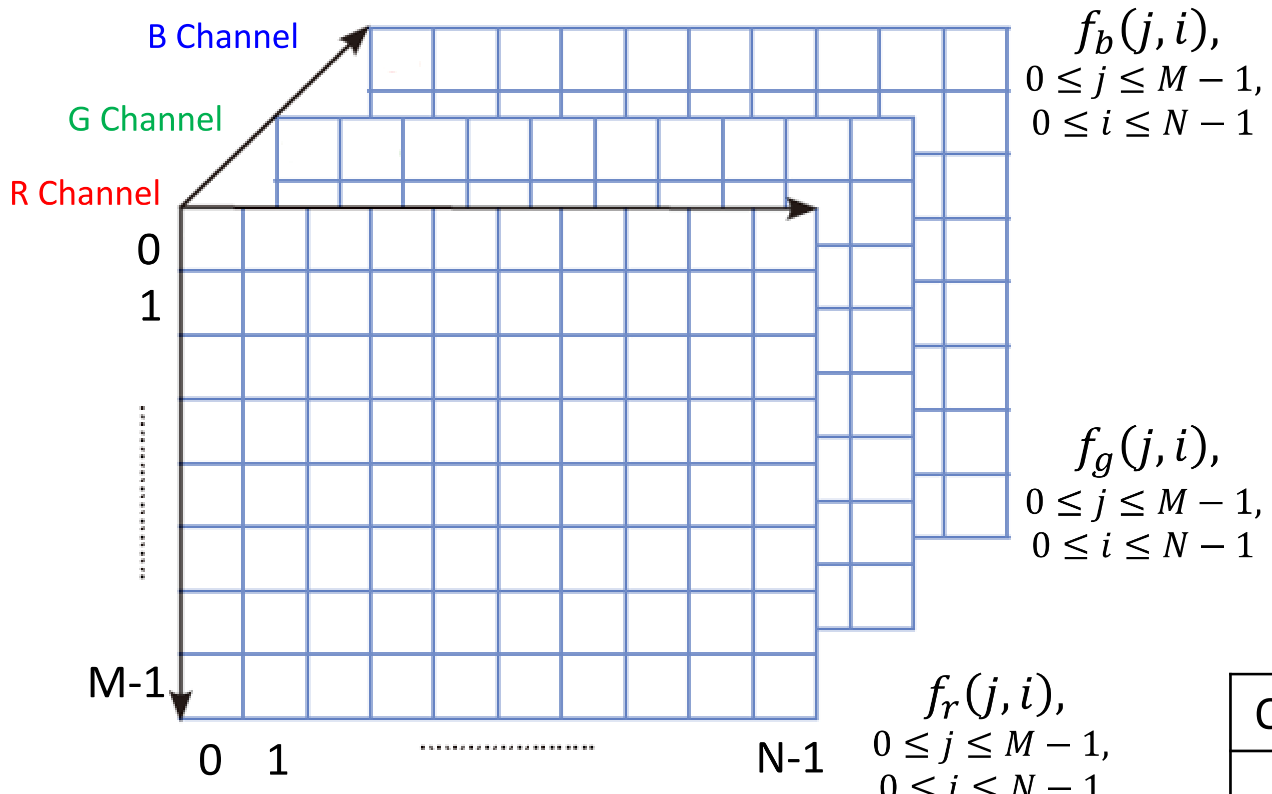
- Modern digital cameras use alternating sensors covered with different colored filters
- Bayer pattern: ½ green + ¼ red + ¼ blue
- Missing colors are interpolated



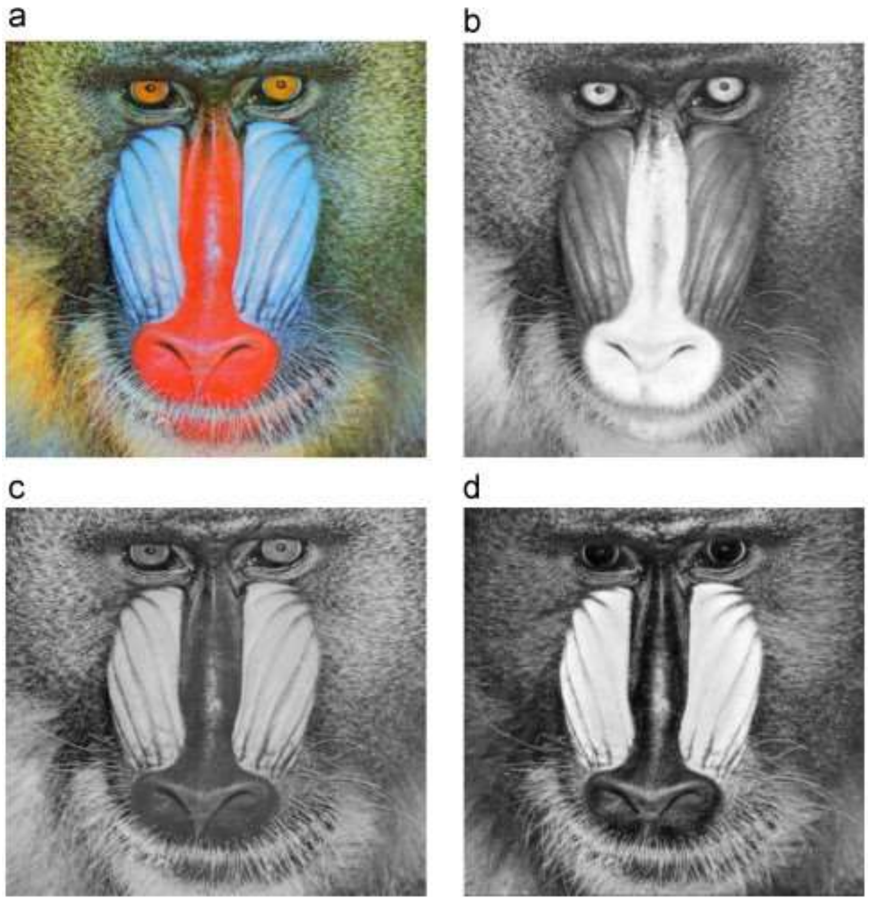


RGB Color Model

- Image representation with RGB color model
  - Represent color image with three channels f_r, f_g, f_b



Original	R
G	B



YC_rC_b/YUV/YIQ Color Model

- YCrCb
  - MPEG/JPEG
  - Y: Luminance
  - C_r,C_b: Chrominance (color info)

$Y = 0.3R + 0.59G + 0.11B$
$C_r = (R-Y)$
$C_b = (B-Y)$

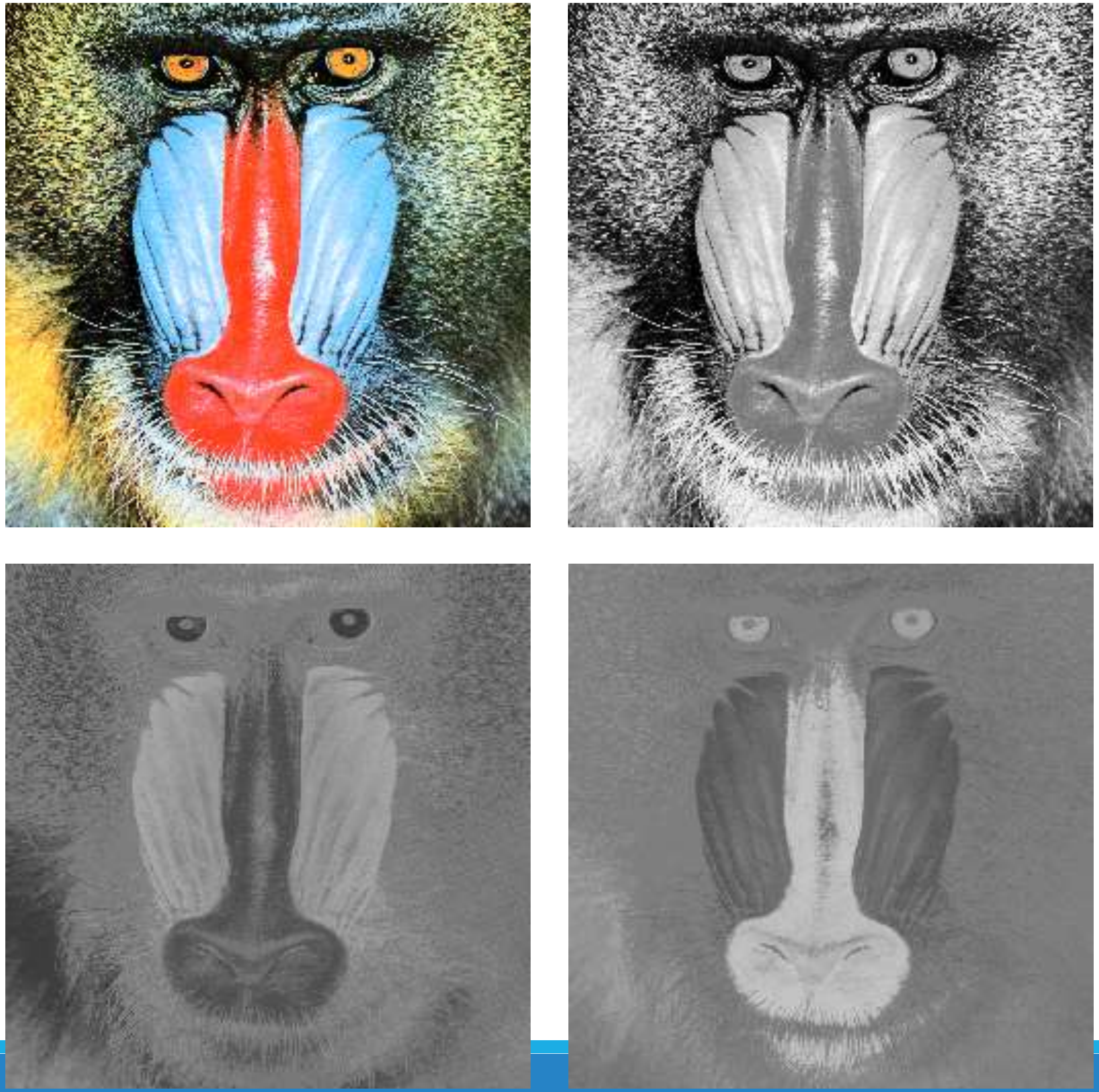
- YUV
  - PAL encoding for Color TV
  - Y: Luminance
  - U,V: Chrominance (color info)

$Y = 0.3R + 0.59G + 0.11B$
$U = (B-Y)\times 0.493$
$V = (R-Y)\times 0.877$

- YIQ
  - NTSC encoding for Color TV
  - Y: Luminance
  - I,Q: Chrominance (color info)

$Y = 0.30R + 0.59G + 0.11B$
$I = 0.60R - 0.28G - 0.32B$
$Q = 0.21R - 0.52G + 0.31B$

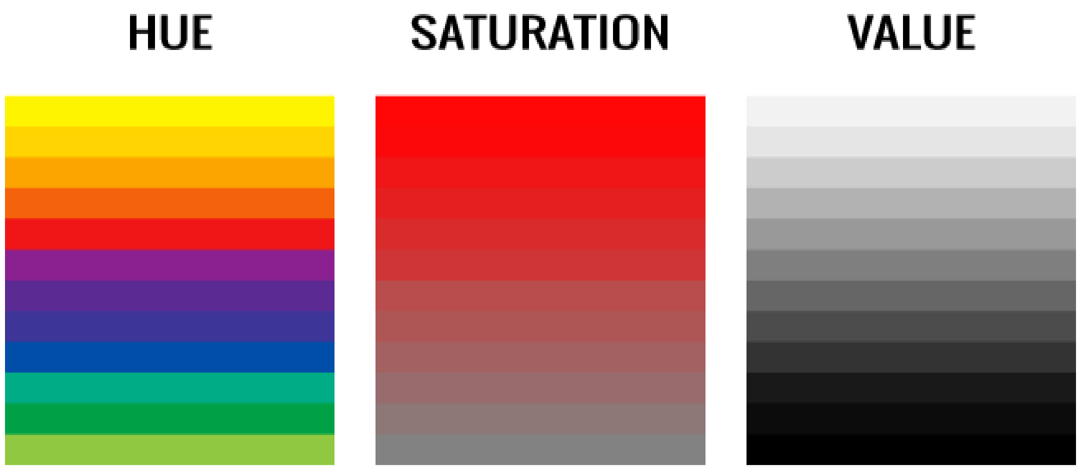
YC_rC_b/YUV/YIQ Color Model

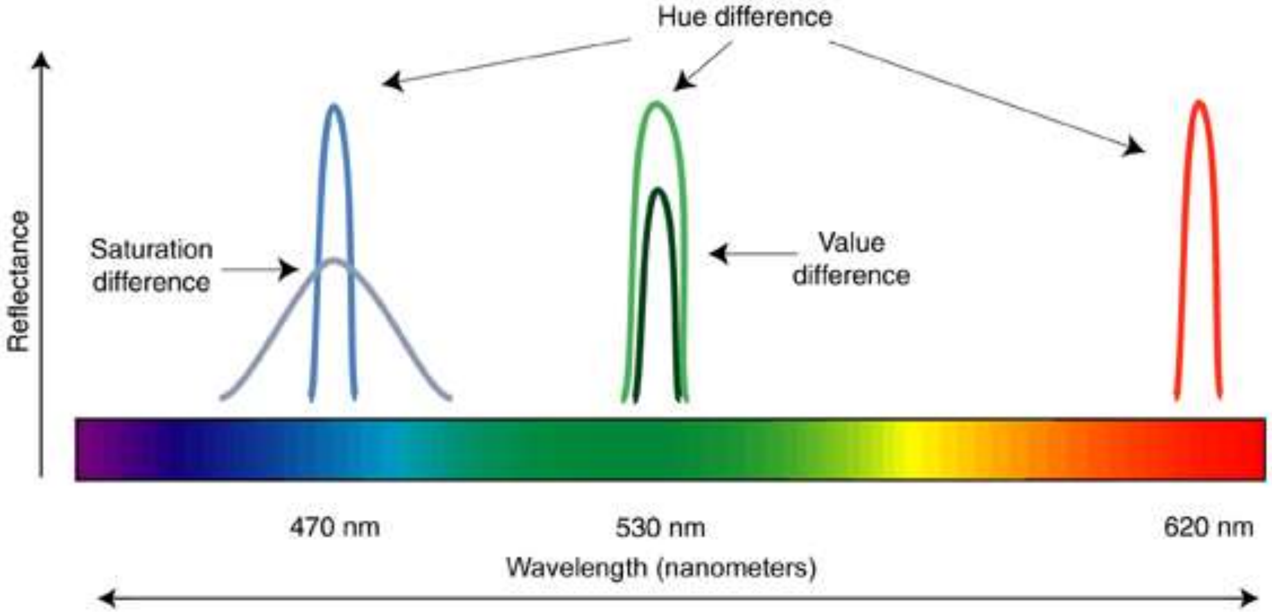


Original	Y
U	V

Properties of Light

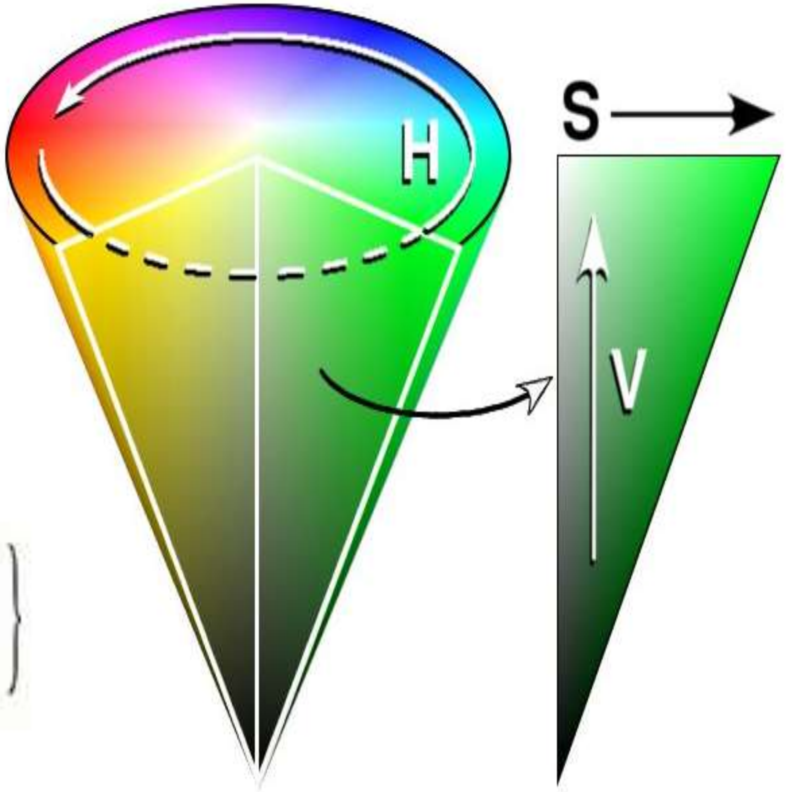
Perception (Qualitative)	Colorimetry (Quantitative)
Hue	Dominant wavelength
Saturation	Purity (Bandwidth)
Value(Brightness)	Luminance (Amount of energy)





HSV Color Model

- HSV (HSB) Model
  - Based on human perception
  - Hue, Saturation, Value
  - Cylindrical coordinate



$$H=\begin{cases}\theta & \text{if } B\le G\\ 360-\theta & \text{if } B\ge G\end{cases}$$

$$\theta=\cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}}\right\}$$

$$S=1-\frac{3}{(R+G+B)}[\min(R,G,B)]$$

$$I=\frac{1}{3}(R+G+B)$$

Contents

- Image
- Color models
- **Image processing operators**
  - Point operators
  - Area-based operators
  - Geometric transformations

Image Processing

- Image processing operators: map pixel values from one image to another
  1. Point operators: manipulate each pixel independently
  2. Neighborhood (area-based) operators: each pixel depends on a small neighboring input values
  3. Geometric transformation: global operation such as rotations, shears, and perspective deformations

Point Operators

- Each pixel output value depends on its input pixel value
- $f_{out}(j,i) = t(f_1(j,i), f_2(j,i), \dots, f_k(j,i))$ (*k*: number of images)

  $f_{out}(\boldsymbol{x}) = t(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \dots, f_k(\boldsymbol{x}))$ (*k*: number of images)

- $f_{out}(\boldsymbol{x}) = af(\boldsymbol{x}) + b$ ($a > 0$: gain, b: bias)
- $f_{out}(\boldsymbol{x}) = a(\boldsymbol{x})f(\boldsymbol{x}) + b(\boldsymbol{x})$ (e.g. graded density filter)

$\Rightarrow$ Linear operation as $t(f_0 + f_1) = t(f_0) + t(f_1)$

Point Operators

- Examples of linear point operators

$$f_{out}(j,i) = t\big(f(j,i)\big)$$

$$= \begin{cases} \min(f(j,i)+a, L-1), \text{(brighter)} \\ \max(f(j,i)-a, 0)\,, \text{(darker)} \\ (L-1) - f(j,i), \text{(inversion)} \end{cases}$$

(a) Original

(b) Brighter (a=32)

(c) Darker (a=32)

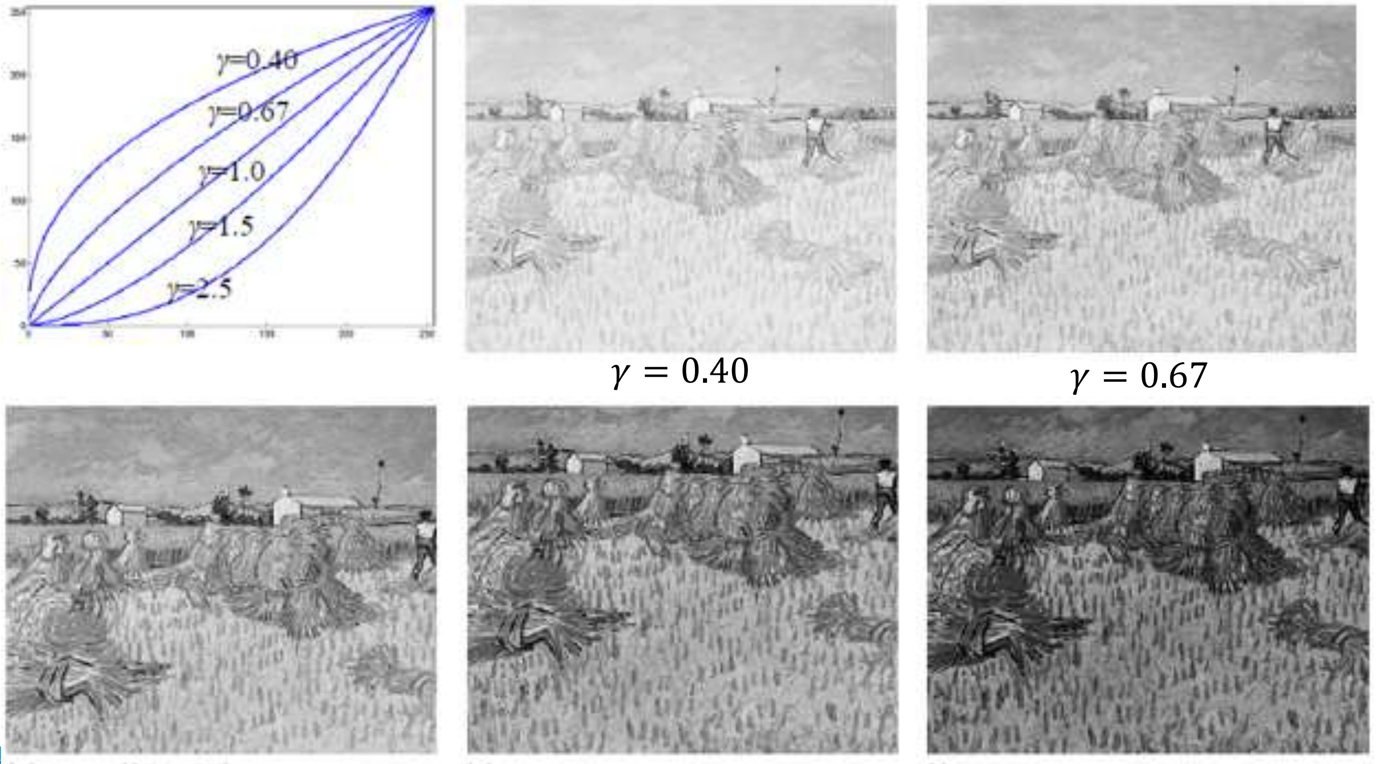
(d) Inversion

Point Operators

- Gamma correction: nonlinear pointwise operation
  - The input voltage of TV/(CRT) monitor and the resulting brightness has a non-linear relationship – characterized by γ $(\mathrm{B} = \mathrm{V}^{\gamma})$
  - Gamma correction adjusts brightness of television or (CRT) monitors

$$f_{out}(j,i) = (L-1) \times \left(\hat{f}(j,i)\right)^{\gamma},$$

$$\text{where } \hat{f}(j,i) = \frac{f(j,i)}{(L-1)}$$



$\gamma = 0.40$ $\gamma = 0.67$

$\gamma = 1.0$ (original) $\gamma = 1.5$ $\gamma = 2.5$

Point Operators

- Linear blend operator: perform *scene-dissolve* between two images (or videos)

$$f_{out}(j,i) = \alpha f_1(j,i) + (1-\alpha) f_2(j,i)$$

Point Operators

- Compositing and matting
  - Matting: extract a f*oreground* object out of a scene
  - Compositing: put the extracted object on top of another *background*
  - Alpha-matted color image: RGB image augmented with A(alpha) channel for transparency(opacity)

$$C = (1 - \alpha)B + \alpha F$$

(a) (b) (c) (d)

Figure 3.4 *Image matting and compositing (Chuang, Curless et al. 2001) © 2001 IEEE: (a) source image; (b) extracted foreground object F; (c) alpha matte α shown in grayscale; (d) new composite C.*

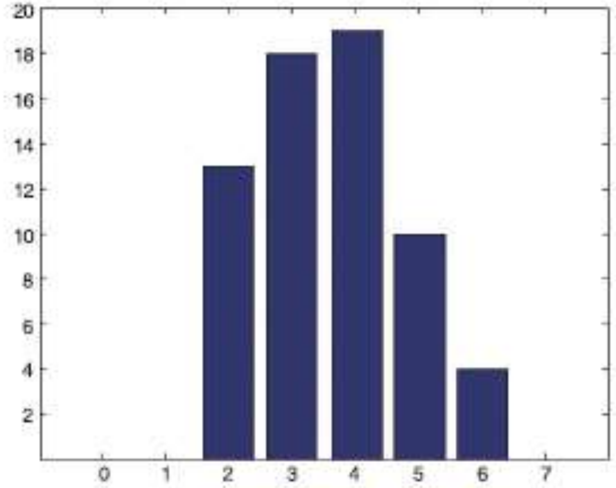
Point Operators: Histogram Equalization

- What is histogram?
  - Count the total number of pixels for each brightness value in [0, L-1]
  - Normalized histogram: divide each histogram value by the total number of pixels

$$h(l) = |\{(j,i) | f(j,i) = l\}|, \text{where } l \in [0, L-1]$$

$$\hat{h}(l) = \frac{h(l)}{M \times N}$$

3	2	2	2	2	3	3	4
3	2	2	2	3	4	3	3
4	3	3	4	4	4	3	3
5	4	4	4	5	4	3	3
5	4	3	4	5	4	3	2
6	5	4	4	5	4	3	2
6	6	5	5	4	3	2	2
6	5	4	5	4	3	2	2



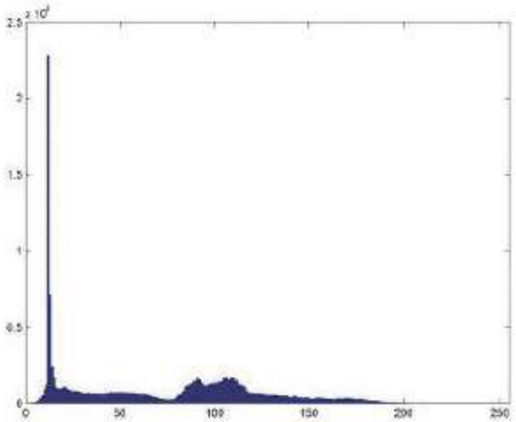
E.g. $M = N = 8, \quad L = 8$

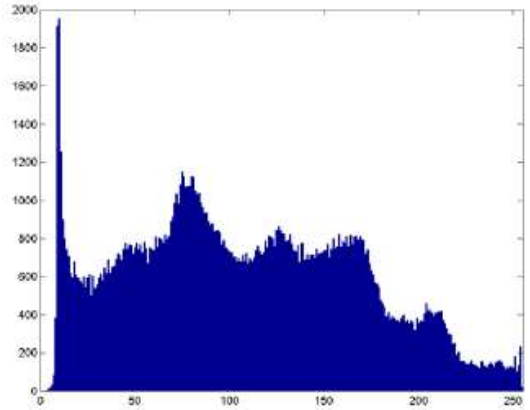
$h(l) = \{0, 0, 13, 18, 19, 10, 4, 0\}$

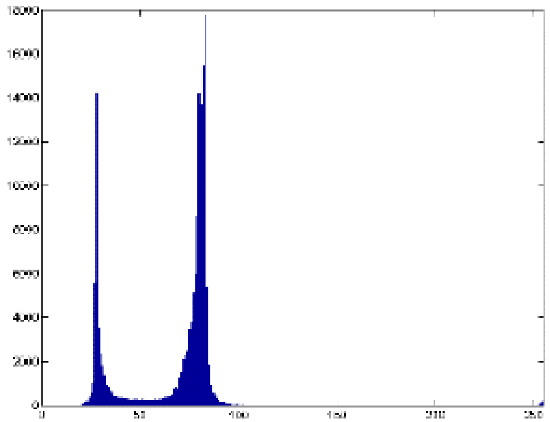
$\hat{h}(l) = \{0, 0, 0.203, 0.281, 0.297, 0.156, 0.063, 0\}$

Histogram

- What can histogram tell us about an image?







Histogram Equalization

- Find a mapping function f(I) to make histogram flat
- Improve the visual appearance of image by increasing the dynamic range of colors
- Use cumulative histogram as a mapping function

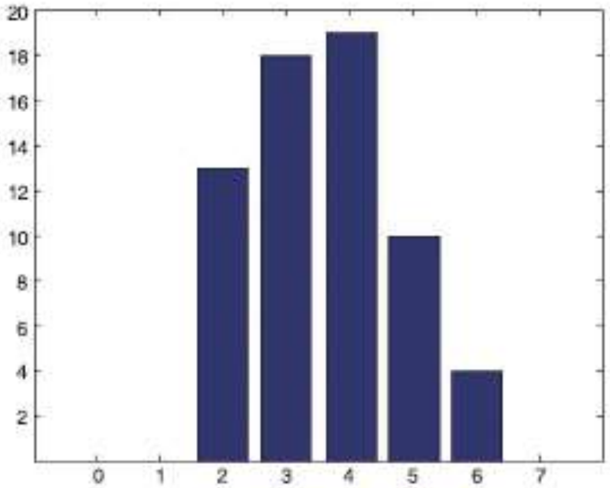
$$l_{out} = T(l_{in}) = round\big(c(l_{in}) \times (L-1)\big), \text{where } c(l_{in}) = \sum_{l=0}^{l_{in}} \hat{h}(l)$$

Histogram Equalization

$$l_{out} = T(l_{in}) = round\big(c(l_{in}) \times (L-1)\big),$$
$$\text{where } c(l_{in}) = \sum_{l=0}^{l_{in}} \hat{h}(l)$$

Q) Perform histogram equalization on the following image

3	2	2	2	2	3	3	4
3	2	2	2	3	4	3	3
4	3	3	4	4	4	3	3
5	4	4	4	5	4	3	3
5	4	3	4	5	4	3	2
6	5	4	4	5	4	3	2
6	6	5	5	4	3	2	2
6	5	4	5	4	3	2	2



E.g. $M = N = 8, \quad L = 8$

$h(l) = \{0, 0, 13, 18, 19, 10, 4, 0\}$

$\hat{h}(l) = \{0, 0, 0.203, 0.281, 0.297, 0.156, 0.063, 0\}$

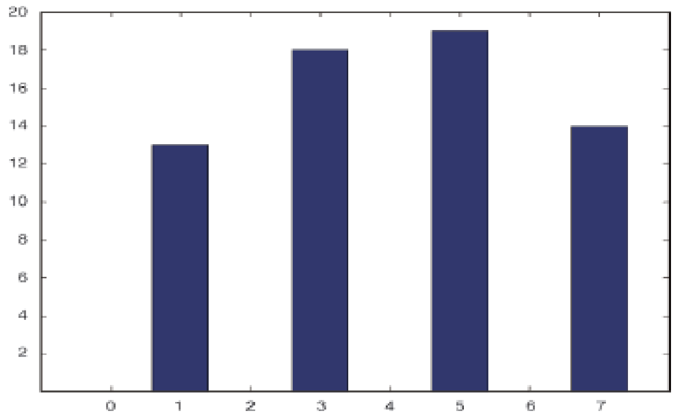
A)

l_{in}	$\hat{h}(l_{in})$	$c(l_{in})$	$c(l_{in}) \times 7$	l_{out}
0	0.0	0.0	0.0	0
1	0.0	0.0	0.0	0
2	0.203	0.203	1.421	1
3	0.281	0.484	3.388	3
4	0.297	0.781	5.467	5
5	0.156	0.937	6.559	7
6	0.063	1.0	7.0	7
7	0.0	1.0	7.0	7

(a) Mapping

3	1	1	1	1	3	3	5
3	1	1	1	3	5	3	3
5	3	3	5	5	5	3	3
7	5	5	5	7	5	3	3
7	5	3	5	7	5	3	1
7	7	5	5	7	5	3	1
7	7	7	7	5	3	1	1
7	7	5	7	5	3	1	1

(b) Image after histogram equalization

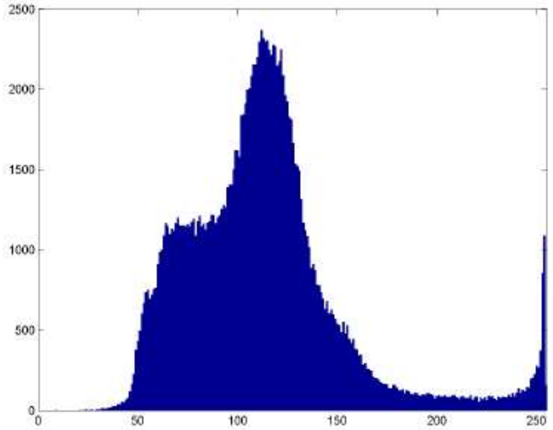


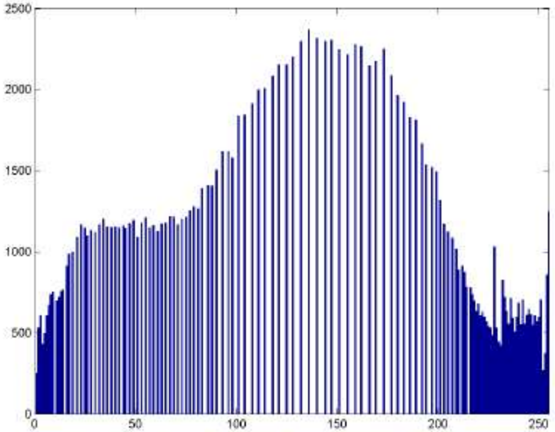
(c) Equalized Histogram

(*) Dynamic range of I widened from [2,6] to [1,7]

Histogram Equalization

Question) Is it always good to perform histogram equalization?





Locally Adaptive Histogram Equalization

- Apply different kinds of equalization in different regions
- Subdivide image into MxM blocks and perform separate histogram equalization

$\Rightarrow$ Problem: block artifacts occur!

$\Rightarrow$ Solution: moving windows (slow) or interpolating transfer functions (Adaptive Histogram Equalization) (not point operators anymore)

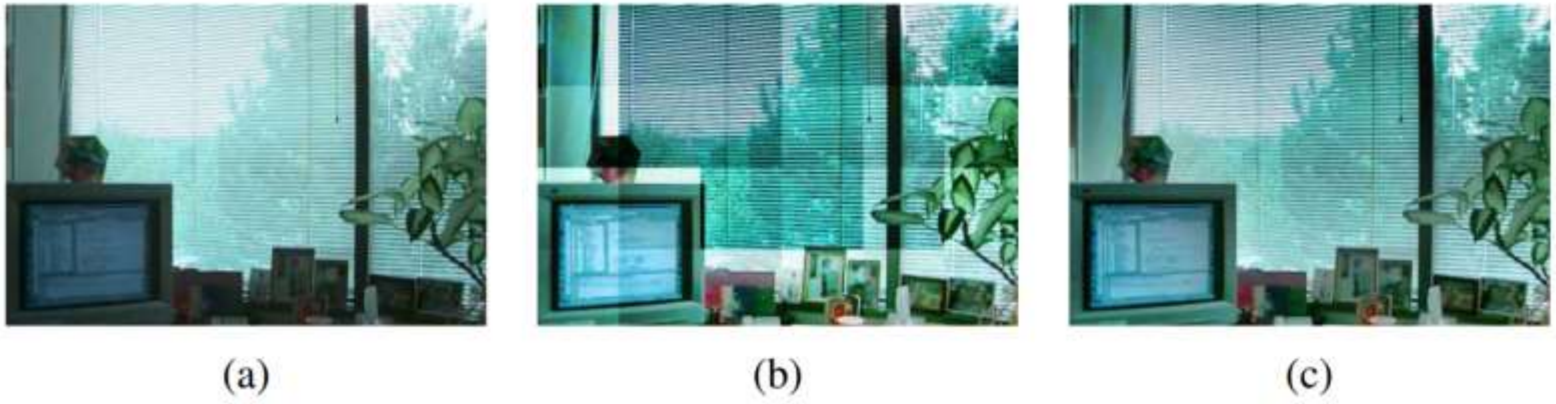


Figure 3.8 *Locally adaptive histogram equalization: (a) original image; (b) block histogram equalization; (c) full locally adaptive equalization.*

Neighborhood (Area-based) Operators

- Neighborhood operator(filter): pixel value computed using a collection of pixel values in a small neighborhood
- Linear filter: a pixel's value is a weighted sum of pixel values within a small neighborhood N
- Correlation and convolution operators

$$\text{1D Correlation: } g(i) = u \otimes f = \sum_{x=-(w-1)/2}^{(w-1)/2} u(x) f(i+x)$$

$$\text{1D Convolution: } g(i) = u \circledast f = \sum_{x=-(w-1)/2}^{(w-1)/2} u(x) f(i-x)$$

$$\text{2D Correlation: } g(i) = u \otimes f = \sum_{x=-(w-1)/2}^{(w-1)/2} \sum_{y=-(h-1)/2}^{(h-1)/2} u(y,x) f(j+y, i+x)$$

$$\text{2D Convolution: } g(i) = u \circledast f = \sum_{x=-(w-1)/2}^{(w-1)/2} \sum_{y=-(h-1)/2}^{(h-1)/2} u(y,x) f(j-y, i-x)$$

Linear Filtering: 1D Example

- Correlation: matching a window with an image
  - u(x): named as a window, a weighted kernel, mask, filter (coefficients)
- Convolution: flipping the window and performing matching
  - u(x): also named as the impulse response ($\because u \circledast \delta(i,j) = u$)

Window u

$u=$	2	4	3

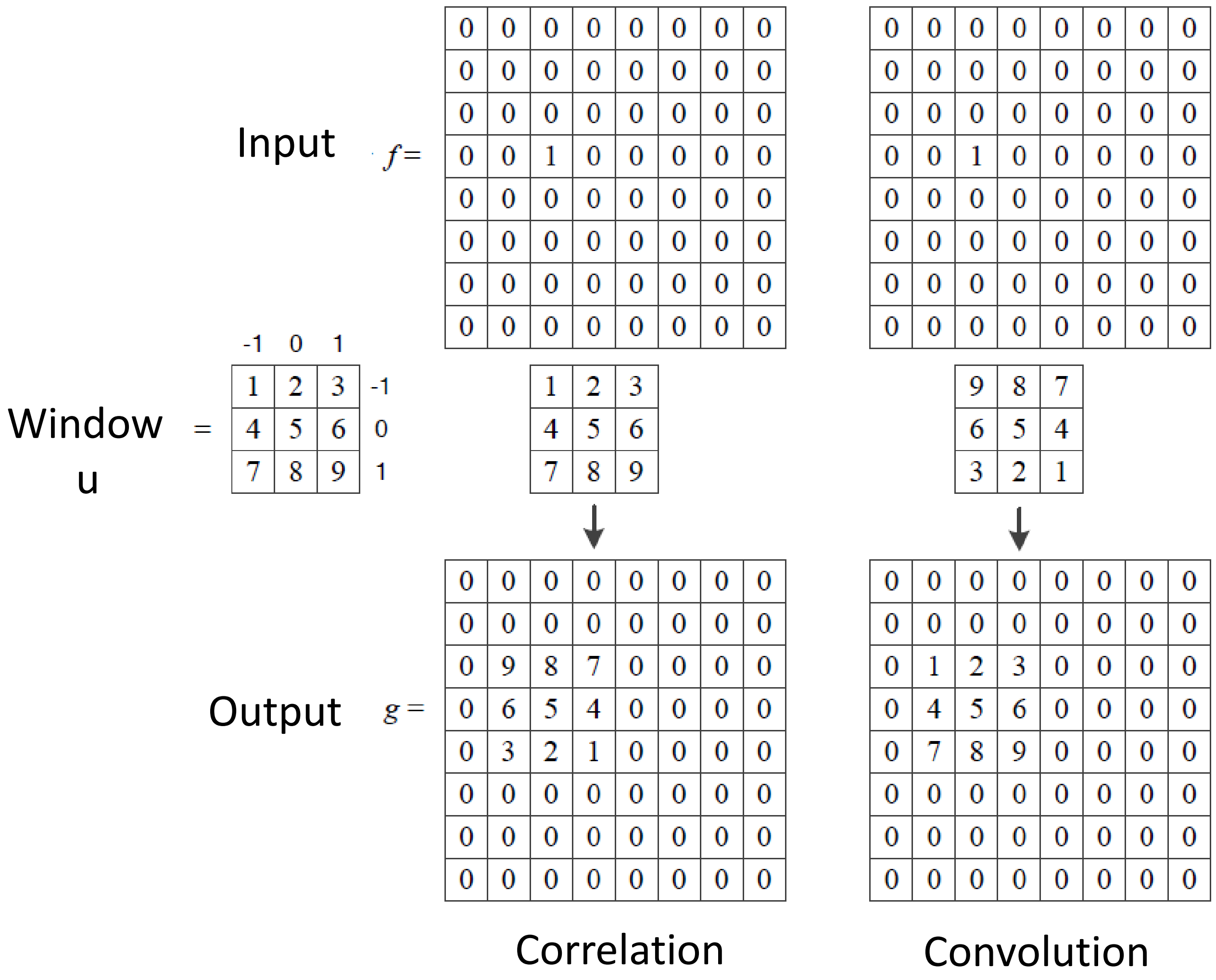
	0	1	2	3	4	5	6	7	8	9
$f=$	0	0	1	0	2	2	4	3	1	0
			×	×	×					
			2	4	3					
				+						
$g=$	-	3	4	8	14	24	29	23	10	-

Correlation

	0	1	2	3	4	5	6	7	8	9
	0	0	1	0	2	2	4	3	1	0
			×	×	×					
			3	4	2					
				+						
	-	2	4	7	12	22	28	26	13	-

Convolution

Linear Filtering: 2D Example



- Here, input f is an impulse response function
- When applying the convolution filter to f, the output image will the same as the window u
- When applying the correlation filter to f, the window will be reversed in the output

Properties of Convolution (Correlation)

- Both correlation and convolution are linear shift-invariant (LSI)
  - The superposition principle: $u \circ (f_0 + f_1) = u \circ f_0 + u \circ f_1$
  - The shift invariance principle:

$$g(j,i) = f(j+k, i+l) \;\Leftrightarrow\; (u \circ g)(j,i) = (u \circ f)(j+k, i+l)$$

$\Rightarrow$ The operator "behaves the same everywhere"

- Correlation and convolution can be written as a matrix-vector multiplication: g = **U**f

$$\boxed{72 \mid 88 \mid 62 \mid 52 \mid 37} * \boxed{1/4 \mid 1/2 \mid 1/4} \quad \Leftrightarrow \quad \frac{1}{4}\begin{bmatrix} 2 & 1 & . & . & . \\ 1 & 2 & 1 & . & . \\ . & 1 & 2 & 1 & . \\ . & . & 1 & 2 & 1 \\ . & . & . & 1 & 2 \end{bmatrix} \begin{bmatrix} 72 \\ 88 \\ 62 \\ 52 \\ 37 \end{bmatrix}$$

Figure 3.12 *One-dimensional signal convolution as a sparse matrix-vector multiplication,*

Separable Filtering

- General convolution filters: requires K^2 operations per pixel
- Separable convolution filter: perform 1D horizontal convolution followed by 1D vertical convolution $\Longrightarrow 2K$ operations per pixel
  - Represent 2D kernel $\mathbf{K} = \mathbf{v}\mathbf{h}^{\mathrm{T}}$
  - Examples of separable filters: box, bilinear, Gaussian, Sobel, LOG

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \quad \frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \quad \frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \quad \frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array}$$

$$\frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \quad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} \quad \frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \quad \frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array} \quad \frac{1}{2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline \end{array}$$

(a) box, $K = 5$ (b) bilinear (c) "Gaussian" (d) Sobel (e) corner

Examples of Linear Filtering

- (a) Box filter (moving average filter): averages the pixels in a K x K window
- (b) Bilinear filter (Bartlett filter): smooths image with a piecewise “tent” function
- (c) Gaussian filter: made by convolving the linear tent function with itself
- (d) Sobel filter: simple 3x3 edge extractor

  (combination of horizontal central difference and a vertical tent filter)
- (e) Simple corner detector: look for

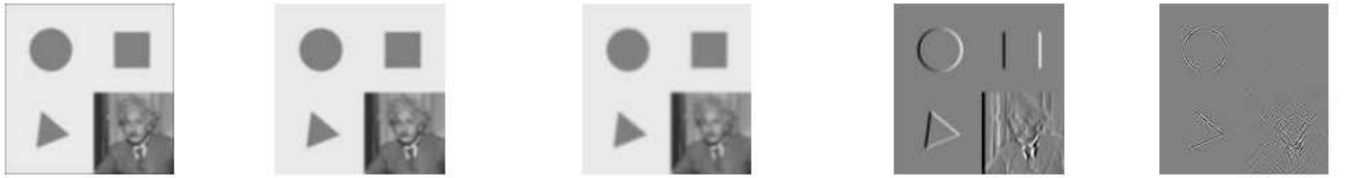
  simultaneous horizontal/vertical

  second derivatives

  (+diagonal edges)

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \quad \frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \quad \frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \quad \frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array}$$

$$\frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \quad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} \quad \frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \quad \frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array} \quad \frac{1}{2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline \end{array}$$



(a) box, $K = 5$ (b) bilinear (c) “Gaussian” (d) Sobel (e) corner

Examples of Linear Filtering

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

(A)

.0000	.0000	.0002	.0000	.0000
.0000	.0113	.0837	.0113	.0000
.0002	.0837	.6187	.0837	.0002
.0000	.0113	.0837	.0113	.0000
.0000	.0000	.0002	.0000	.0000

(B)

0	-1	0
-1	5	-1
0	-1	0

(C)

Original

1	1	1
0	0	0
-1	-1	-1

(D)

1	0	-1
1	0	-1
1	0	-1

(E)

.0304	.0501	0	0	0
.0501	.1771	.0519	0	0
0	.0519	.1771	.0519	0
0	0	.0519	.1771	.0501
0	0	0	.0501	.0304

(F)

Box

Gaussian

Sharpening

Horizontal Edge

Vertical Edge

Motion

Summed Area Table (Integral Image)

- Accelerate convolution when box filters of different sizes are used
- Summed area table: $s(j,i) = \sum_{k=0}^{j} \sum_{l=0}^{i} f(k,l)$ or

$$s(j,i) = s(j-1,i) + s(j,i-1) - s(j-1,i-1) + f(j,i)$$

- s(j,i) is also called an integral image
- To find the summed area in $[j_0, j_1] \times [i_0, i_1]$, we need 4 samples:

$$S([j_0,j_1],[i_0,i_1]) = s(j_1,i_1) - s(j_1,i_0-1) - s(j_0-1,i_1) + s(j_0-1,i_0-1)$$

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

(a) S = 24

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

(b) s = 28

3	5	12	*14*	17
4	11	19	24	31
9	17	28	38	46
13	24	37	**48**	62
15	30	44	59	81

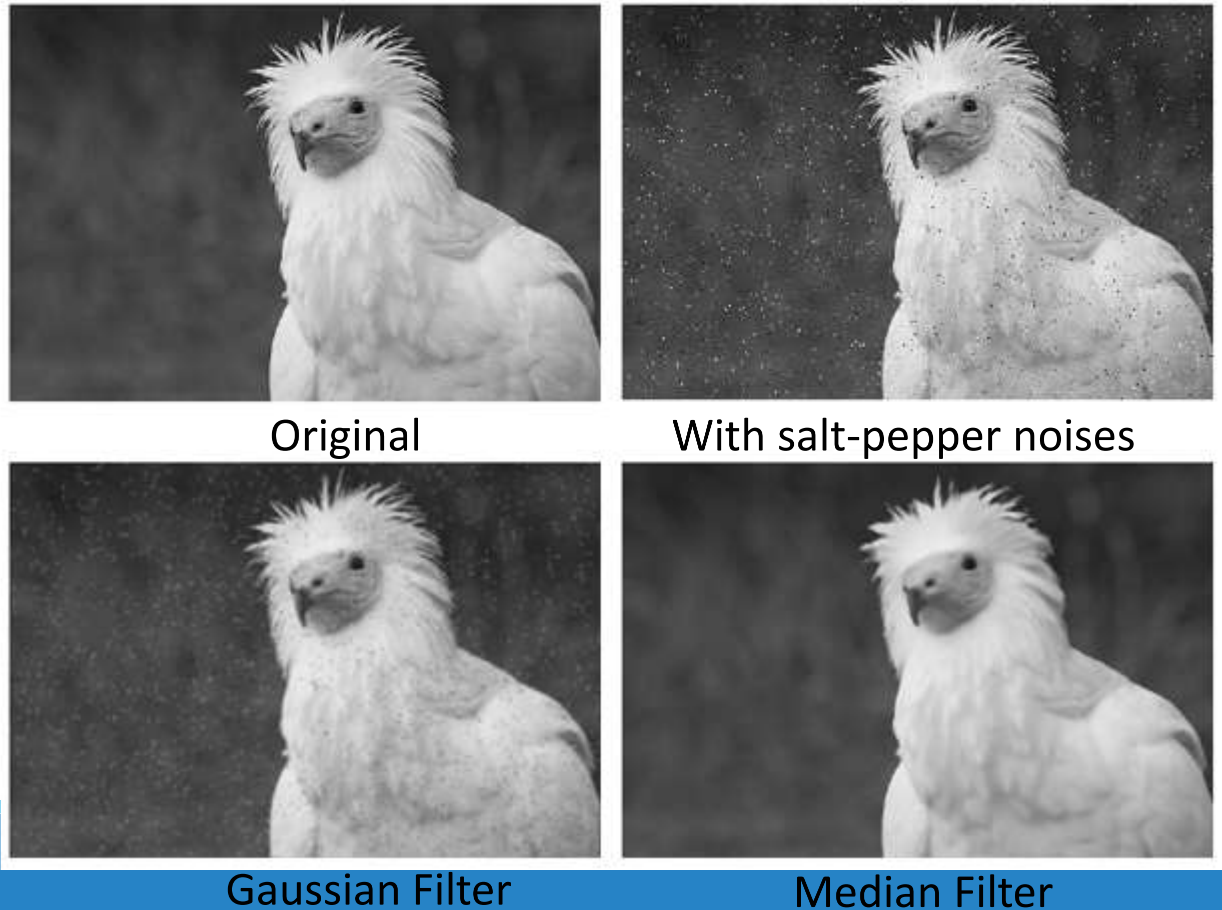
(c) S = 24

(a) Original image
(b) Summed area table
(c) Computation of area sum

Non-Linear Filtering

- Linear filter computes a weighted sum of input pixels
- Non-linear filters perform better in some applications
  - E.g. Edge-preserving filtering, removing shot noises
- Example non-linear filters:

  median filter, bilateral filter



Non-Linear Filtering

- Median filter: selects the median value from each pixel's neighborhood
  - Can be implemented via linear-time algorithm
  - Robust to removing shot noises while preserving edges

1	2	1	2	4
2	1	3	5	8
1	3	7	6	9
3	4	8	6	7
4	5	7	8	9

(a) median = 4

- Bilateral filter: reject pixels whose values differ too much from the central pixel value (in a soft way)

$$g(j,i) = \frac{\sum_{k,l} f(k,l)w(j,i,k,l)}{\sum_{k,l} w(j,i,k,l)}, \text{ where } w(j,i,k,l) = \exp\left(-\frac{(j-k)^2+(i-l)^2}{2\sigma_d^2} - \frac{\|f(j,i)-f(k,l)\|^2}{2\sigma_r^2}\right)$$

w(j,I,k,l): Bilateral weight function

Domain kernel

Data-dependent range kernel

Non-Linear Filtering

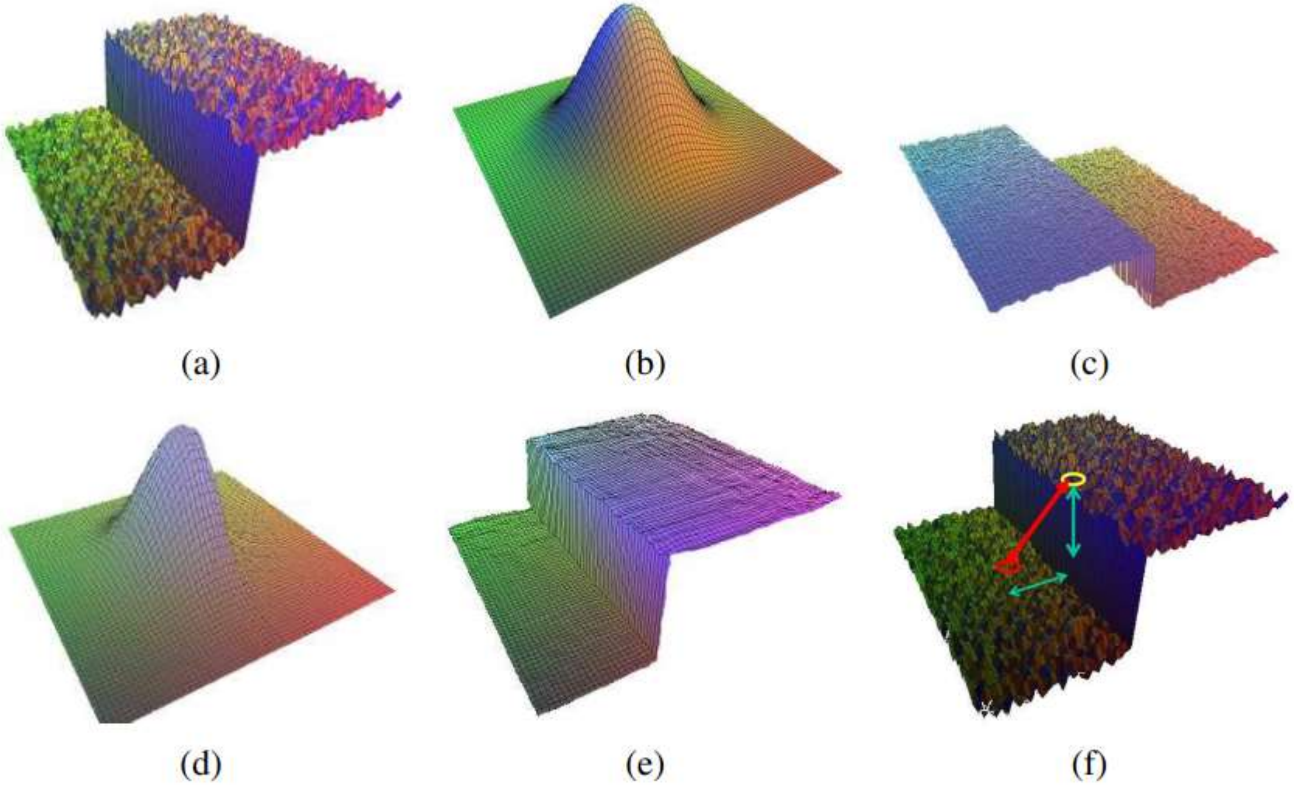


Figure 3.20 *Bilateral filtering (Durand and Dorsey 2002) © 2002 ACM: (a) noisy step edge input; (b) domain filter (Gaussian); (c) range filter (similarity to center pixel value); (d) bilateral filter; (e) filtered step edge output; (f) 3D distance between pixels.*