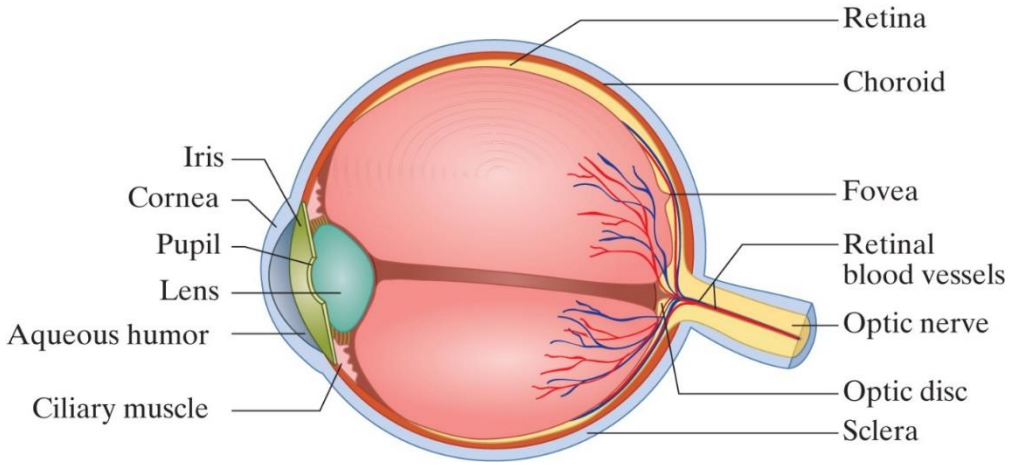
Image enhancement

Image Formation

What is light?
“An electromagnetic radiation within the portion of the electromagnetic spectrum that can be perceived by the human eye”



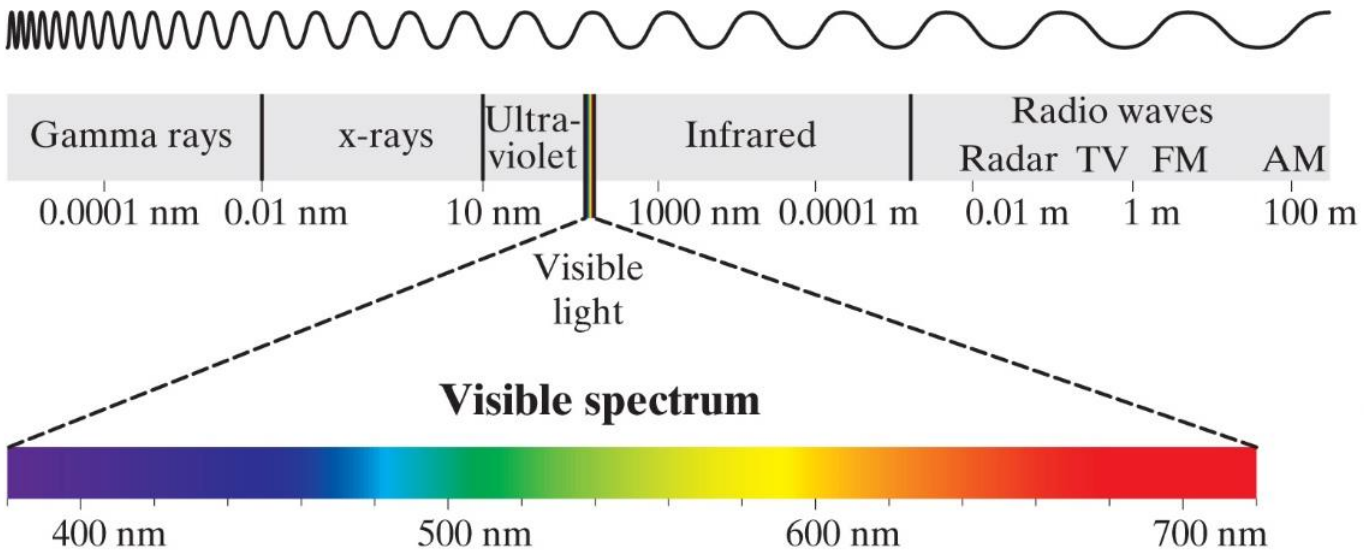
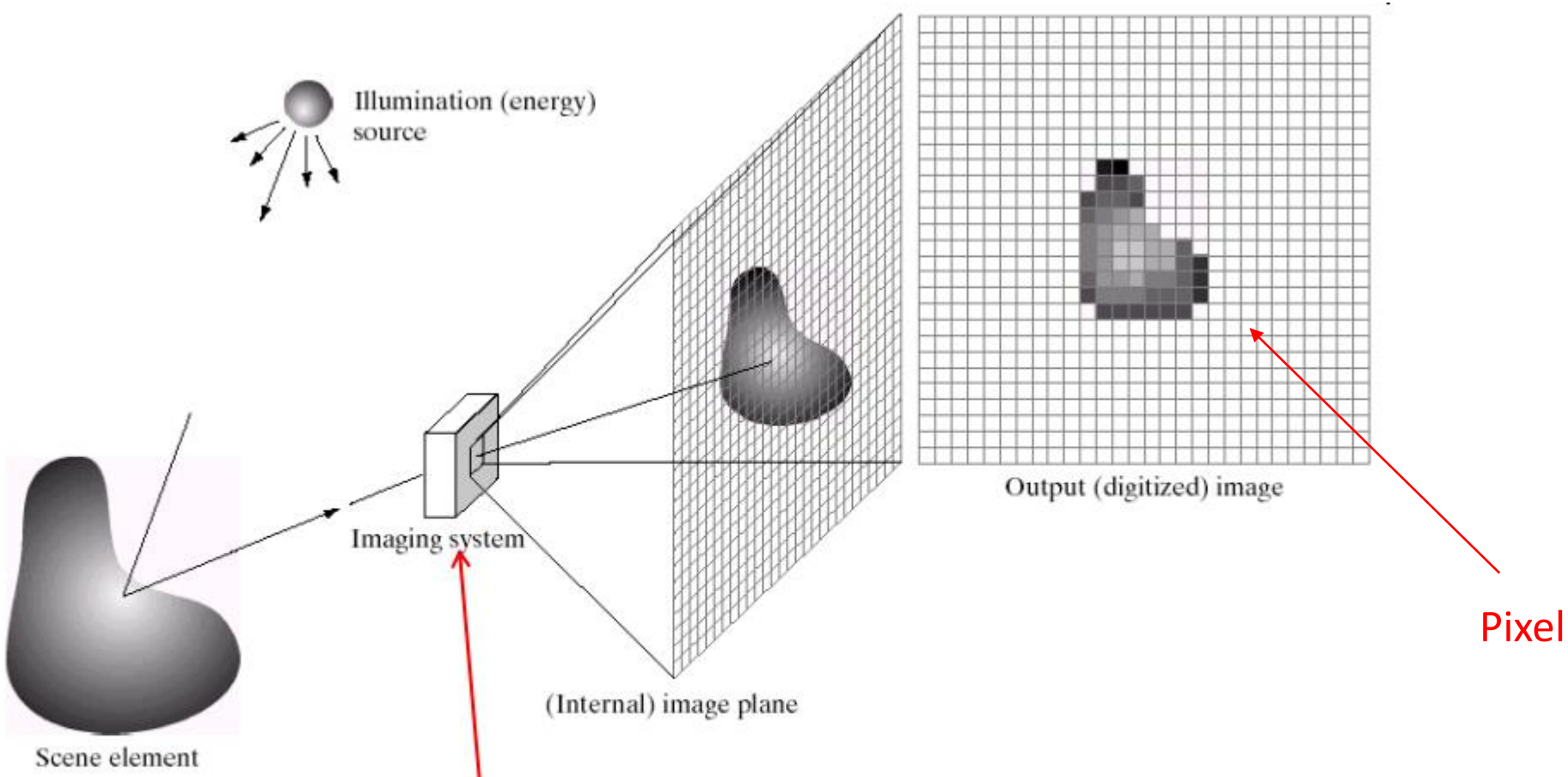
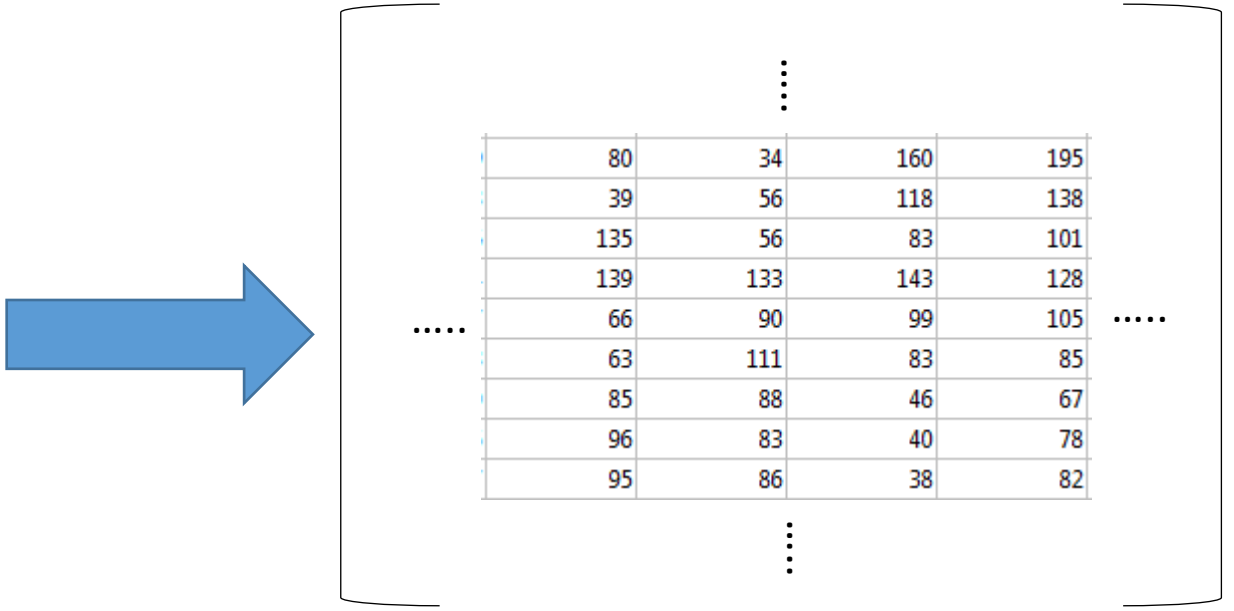


Image Formation



What is an image?

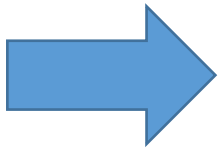
- Mathematically, an image is represented by a 2D matrix *I(r,c)*.
- r: the row number r ∈ $\{1,2,\dots,R\}$, c: column number c ∈ $\{1,2,\dots,C\}$,

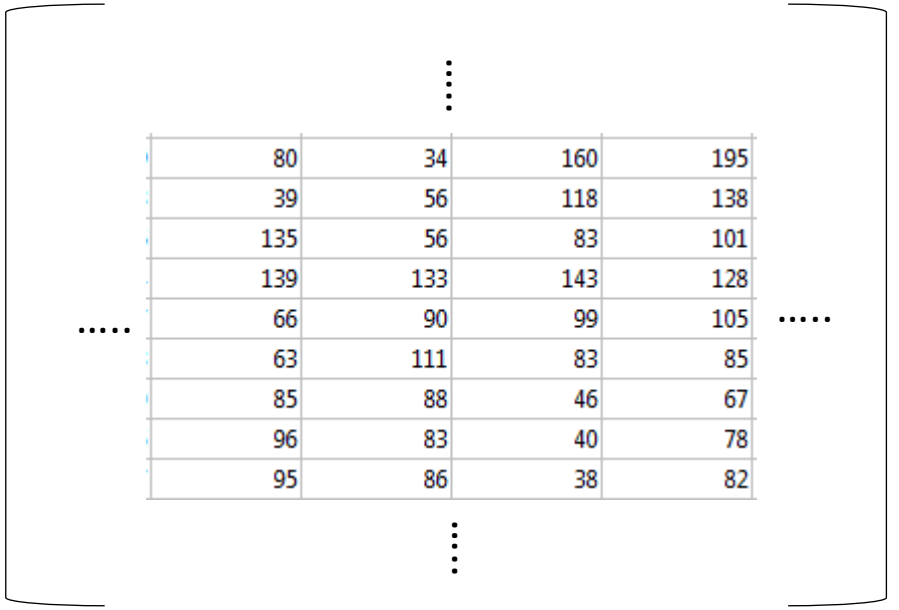


80	34	160	195
39	56	118	138
135	56	83	101
139	133	143	128
66	90	99	105
63	111	83	85
85	88	46	67
96	83	40	78
95	86	38	82

What is an image?

- *I(r, c)* represents a measured value occurring at spatial location *(r,c)*





80	34	160	195
39	56	118	138
135	56	83	101
139	133	143	128
66	90	99	105
63	111	83	85
85	88	46	67
96	83	40	78
95	86	38	82

Image Resolution

- Spatial resolution: Column (C) by Row (R) dimensions of the image.

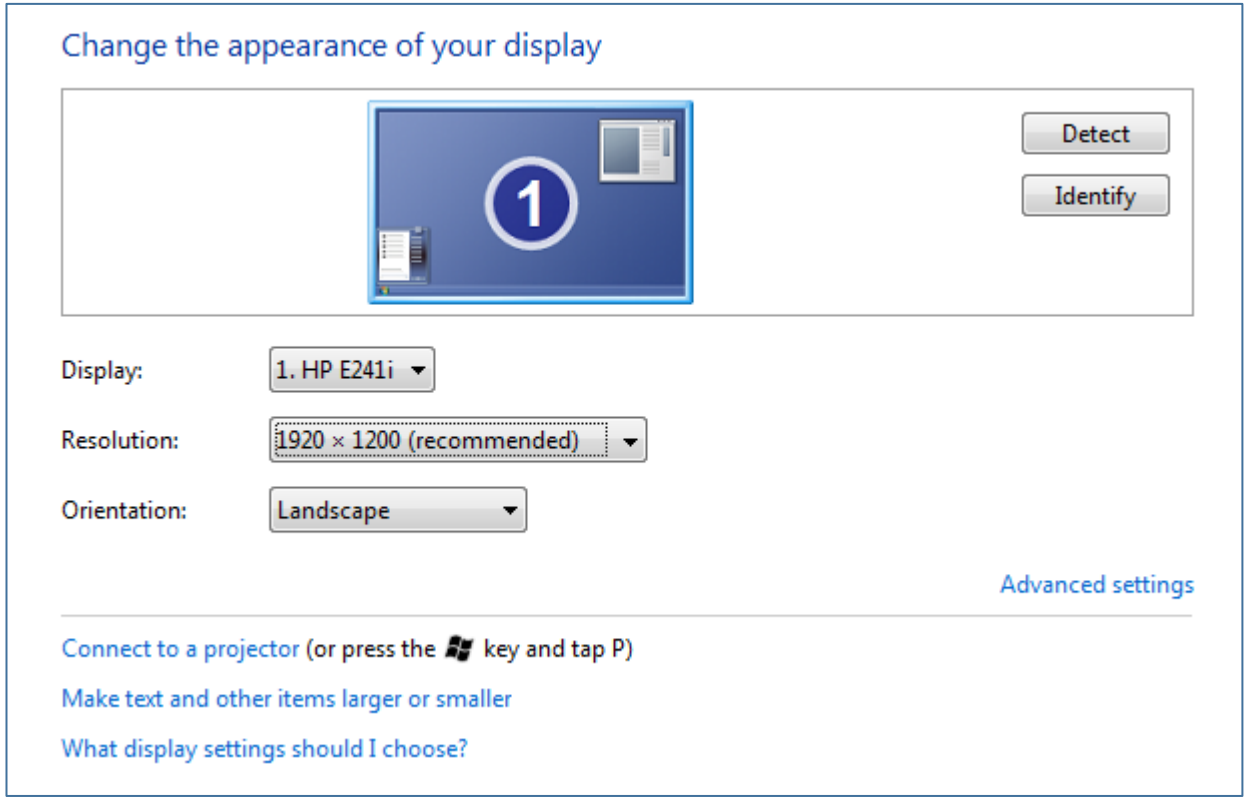


Image Resolution (Quantization)

- Bit resolution: Number of possible values that a pixel may have.
- A binary image is one-bit image (*i.e.* has two colors)

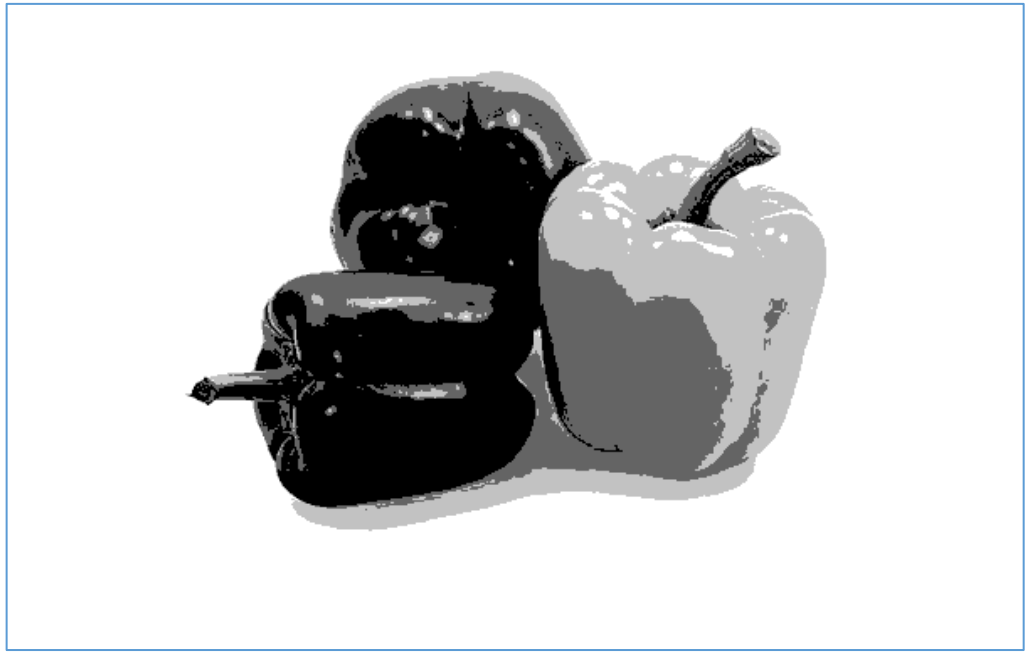
An 8-bit image

A binary image

Image Resolution

- 3-bit images have 2^3 gray scale levels.
- The more intensity levels used, the finer the level of detail discernable in an image.

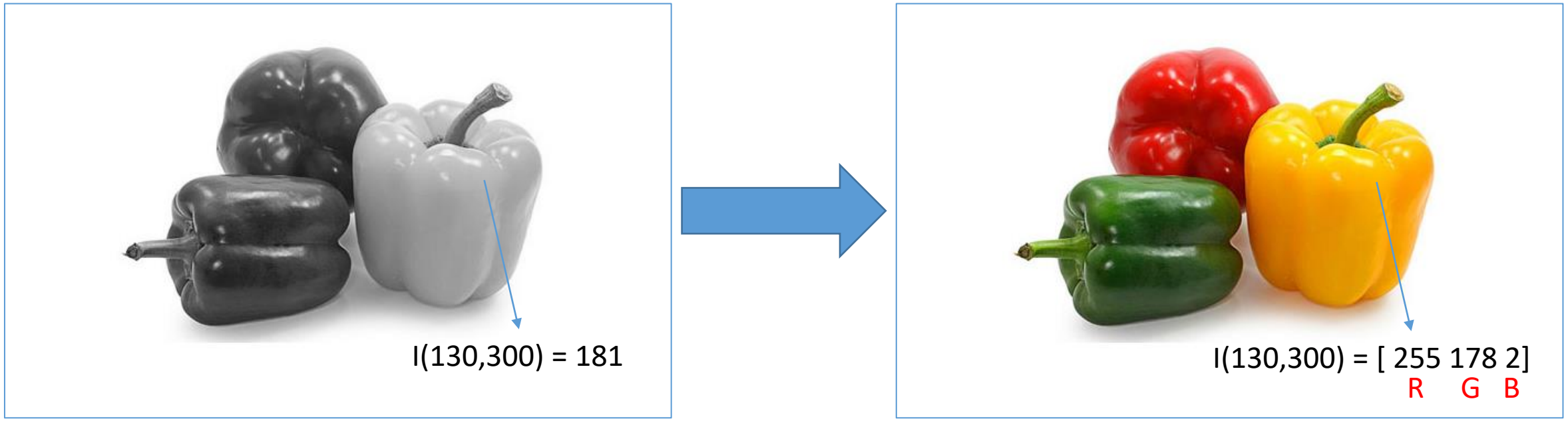
An 8-bit image



A 3-bit image

Color Images

- Color images replace intensity values, used in grey scale images, by a vector of three parameters to represent red, green, and blue color components.



Color and gray scale versions for an 8-bit image of size R = 272 and C= 400

Color Images

- Color images are normally represented as 3D matrices of size R X C X 3.
- Conversion from RGB color value to gray scale intensities

$$Grayscale\ intensity = 0.2989\ X\ R + 0.5870\ X\ G + 0.1140\ X\ B$$

- In Matlab, use the command '*rgb2gray*'

Color Images

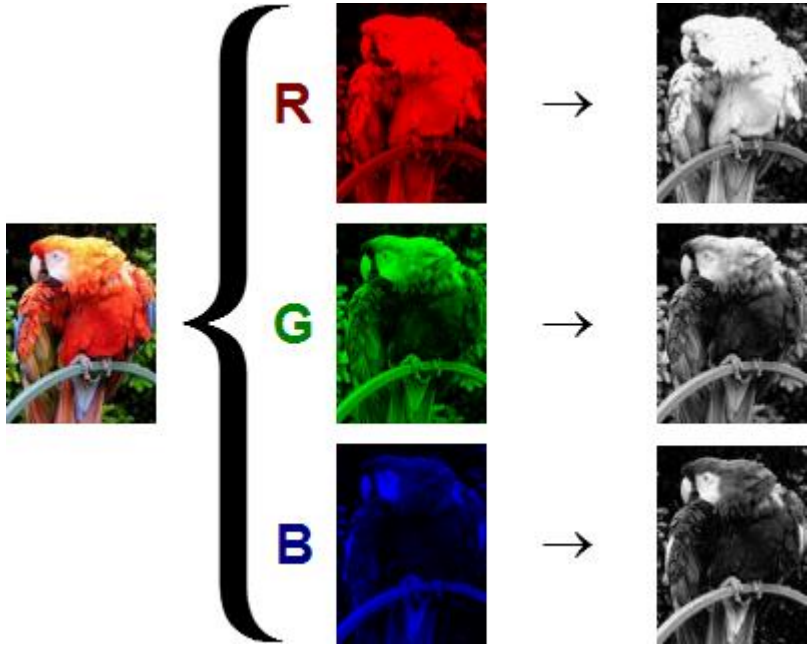
- An 8-bit per channel color image can show 16777216 distinct colors - Why?

  $2^8 \times 2^8 \times 2^8 = 2^{24}$

- How many colors can 1-bit color image show?

  $2 \times 2 \times 2 = 8$

Color Images



- Examples of other color model: the **HSV** (hue-saturation-value)

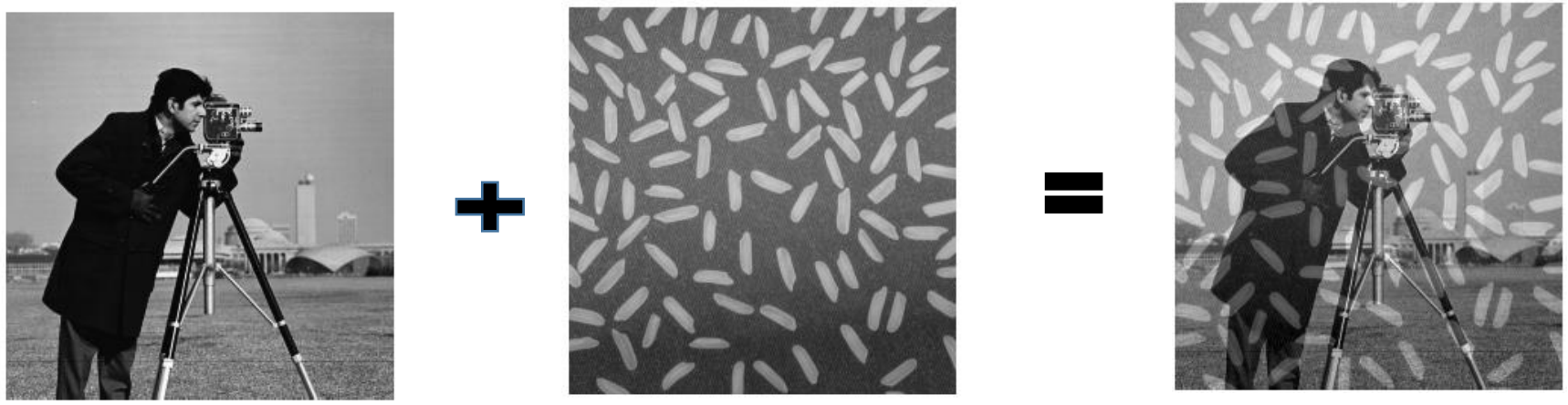
Constant addition

Contrast enhancement by the addition of the mean image intensity value C=54.6

Image addition (blending)

- Blending of two images: Adding intensity values from two images

$$I_{output} = I_A + I_B$$



A *B* *½ A + ½ B*

Image addition (blending)

- In general, linear image addition takes the form

$$I_{output} = \alpha\, I_A + (1 - \alpha) I_B$$

Where α ranges from 0 to 1.

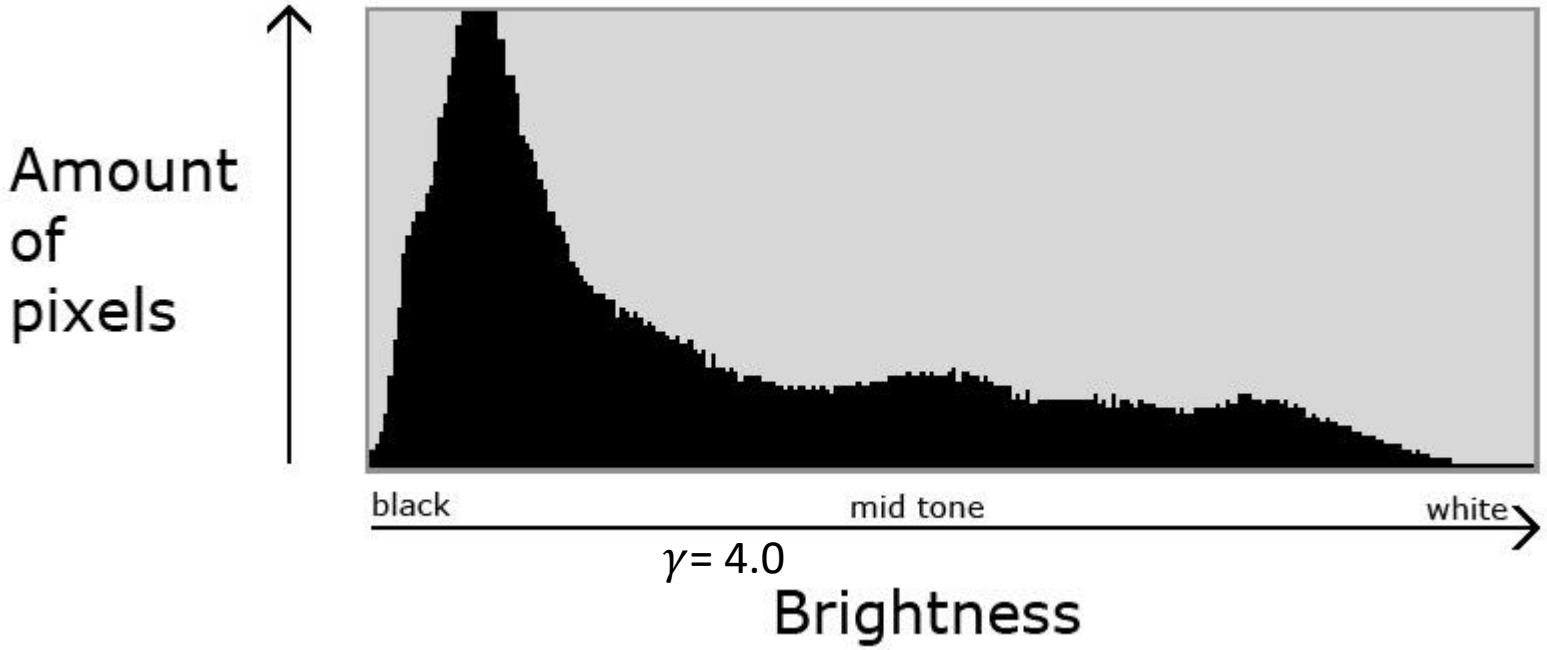
Image subtraction

- Similar to addition. Example applications: contrast adjustment, and detecting change in video sequences.

$$I_{output} = |I_A - C|$$

$$I_{output} = |I_A - I_B|$$

Image Histograms



- An image histogram is a plot of the distribution of image intensity values.
- x-axis: intensity values, y-axis: number of pixels with intensity value of x.
- The *'histogram'* command plots the image histogram.

Image Histograms

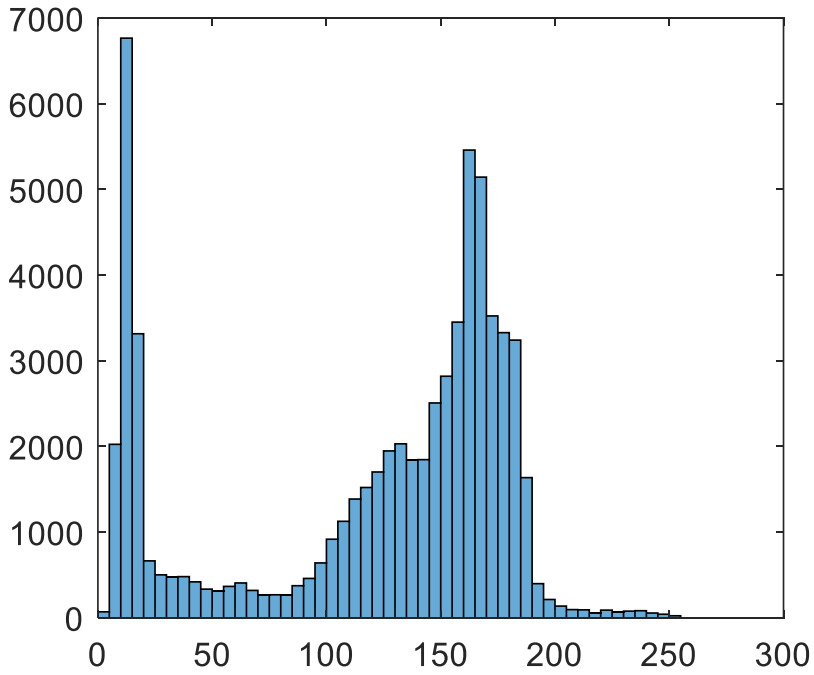
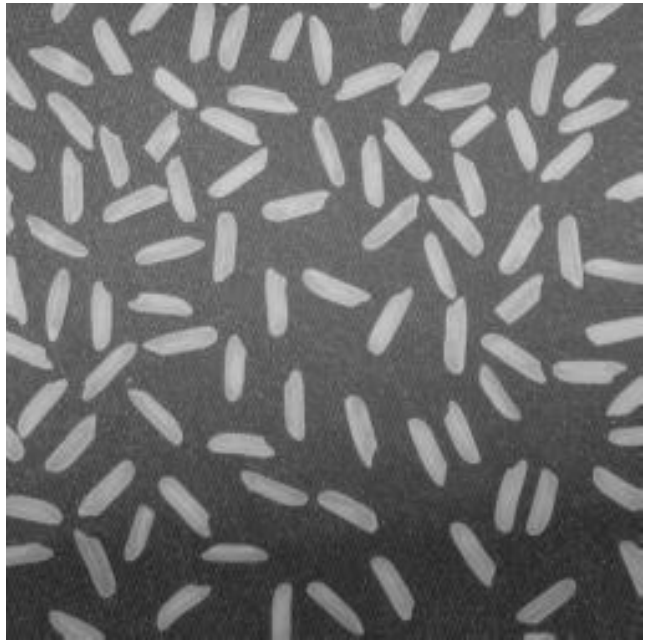
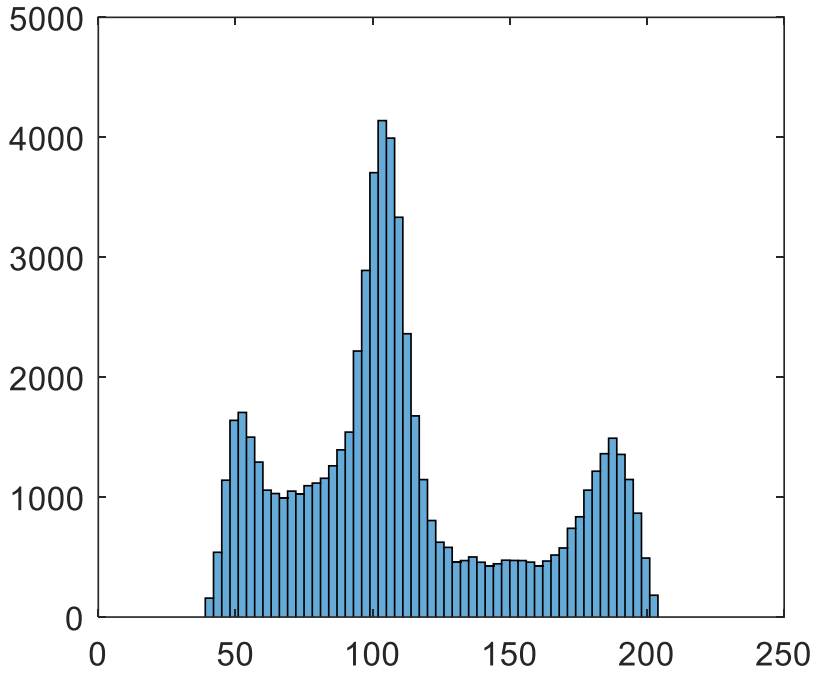
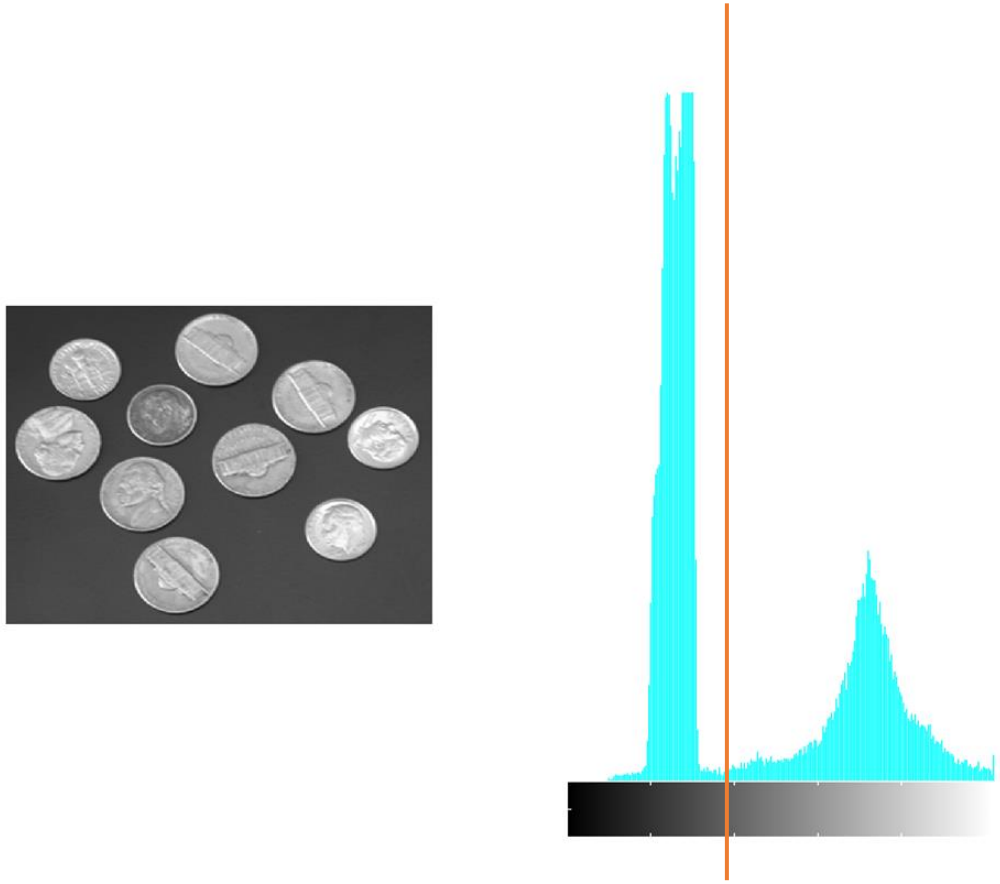


Image Histograms





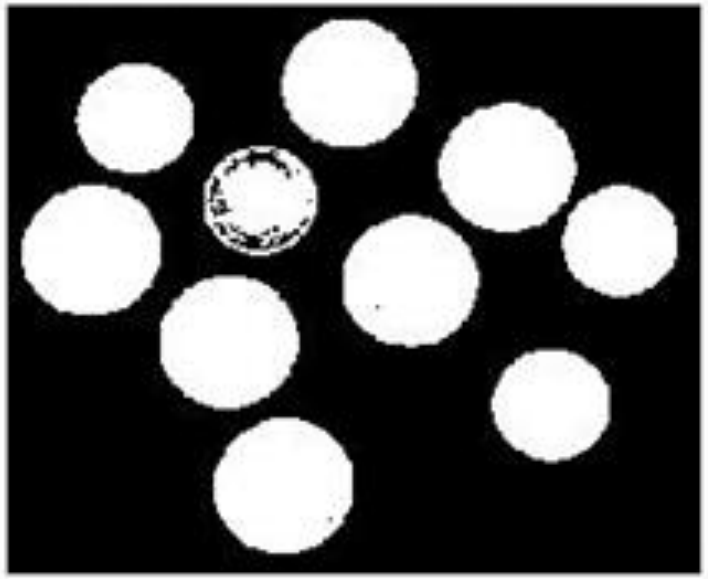
Histograms



A sample image and its histogram

Thresholding

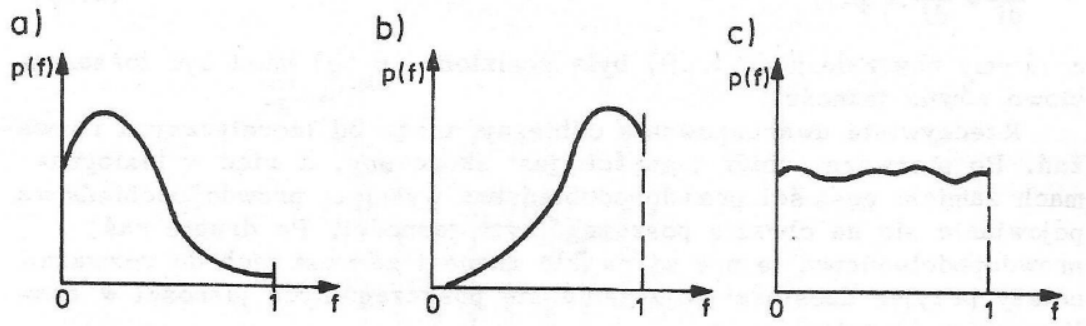
Setting intensity values < Threshold to zero, other values to one.



Histogram equalization

- Goal: to create an image with equal presence of all intensities (well-contrasted images)
- Method: histogram equalization

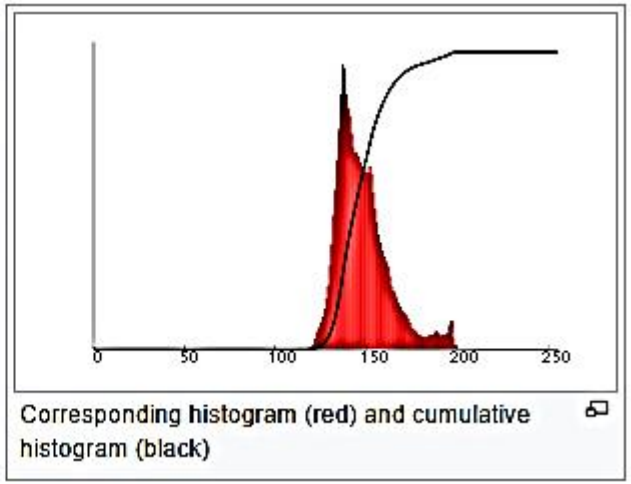
Histograms represent probabilities (relative frequencies) of various intensities within the whole image.



too dark image too bright image Balanced image

Histogram equalization

Before Histogram Equalization



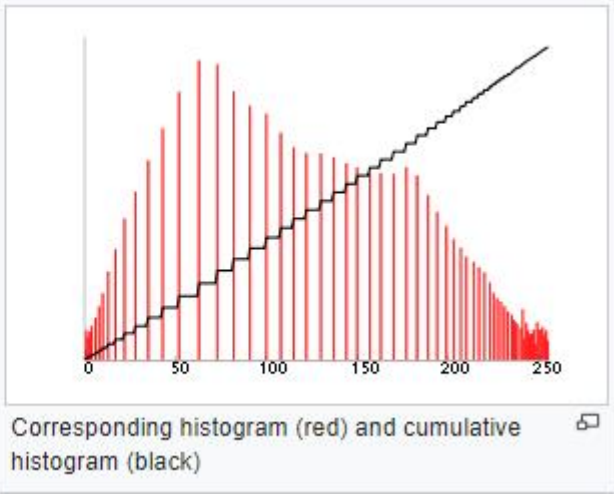
Corresponding histogram (red) and cumulative histogram (black)

- Contrast is low in the image. A look at the histogram reveals that pixel intensity values occupy a short range of allowed range.
- Goals is to enhance the contrast by extending the histogram while having minimal perceptual artifacts in the image

Histogram equalization

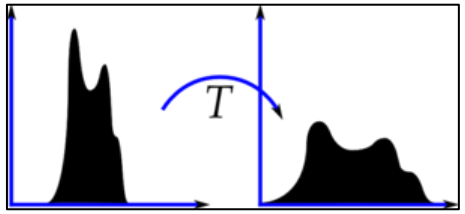
- Histogram equalization maps intensity values to new values such that the cdf is approximately step-wise increasing.
- This is achieved by computing the cdf of each pixel intensity value. Recall that cdf values range between 0 and 1.
- Next, the maximum possible intensity value in the image (i.e. 255) is multiplied by the cdf(i) to obtain the new pixel value, where i is each pixel intensity value.

After Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

Histogram equalization



- Method
  1. Compute the CDF at all intensity values.

$$C(i) = \frac{j}{MN},$$

where *i* is input image intensity value,

j is the number of pixels with intensity values <= i,

MN is the total number of pixels in the image.

2. Compute the output image intensities using

$$I_{output}\,(m,n) = \left[L \times C\,(\,I_{input}(m,n)\,)\right],$$

Round to nearest integer

where *L* is the maximum possible intensity value.

Histogram equalization

- The following distribution of intensities represent a 3-bit image

0	1	3	4
1	2	2	3
1	3	4	4
3	2	5	2

- Do you see issues that can affect the quality of the perceived image?

Histogram equalization

$$I_{output}\,(m,n) = \left[L \times C\,(\,I_{input}(m,n)\,)\right]$$

pdf value cdf value

Maximum possible intensity value

0	1	3	4
1	2	2	3
1	3	4	4
3	2	5	2

(a)

i	$\hat{h}_i$	$\hat{C}_i$	$7\hat{C}_i$
0	1/16	1/16	0
1	3/16	4/16	2
2	4/16	8/16	4
3	4/16	12/16	5
4	3/16	15/16	7
5	1/16	16/16	7
6	0/16	16/16	7
7	0/16	16/16	7

(b)

0	2	5	7
2	4	4	5
2	5	7	7
5	4	7	4

(c)

i	$\hat{h}_i$
0	1/16
1	0/16
2	3/16
3	0/16
4	4/16
5	4/16
6	0/16
7	4/16

(d)

Figure 5.9. Numerical example of histogram equalization: (a) a 3-bit image, (b) normalized histogram and CDF, (c) the equalized image, and (d) histogram of the result.

Reference: Kenny, The Art of Image Processing with Java,2010

Histogram equalization on color images

- The goal of contrast enhancement is normally altering intensities.
- Depending on project needs, applying contrast enhancement to RGB matrices generate undesired artifacts.

Histogram equalization on color images

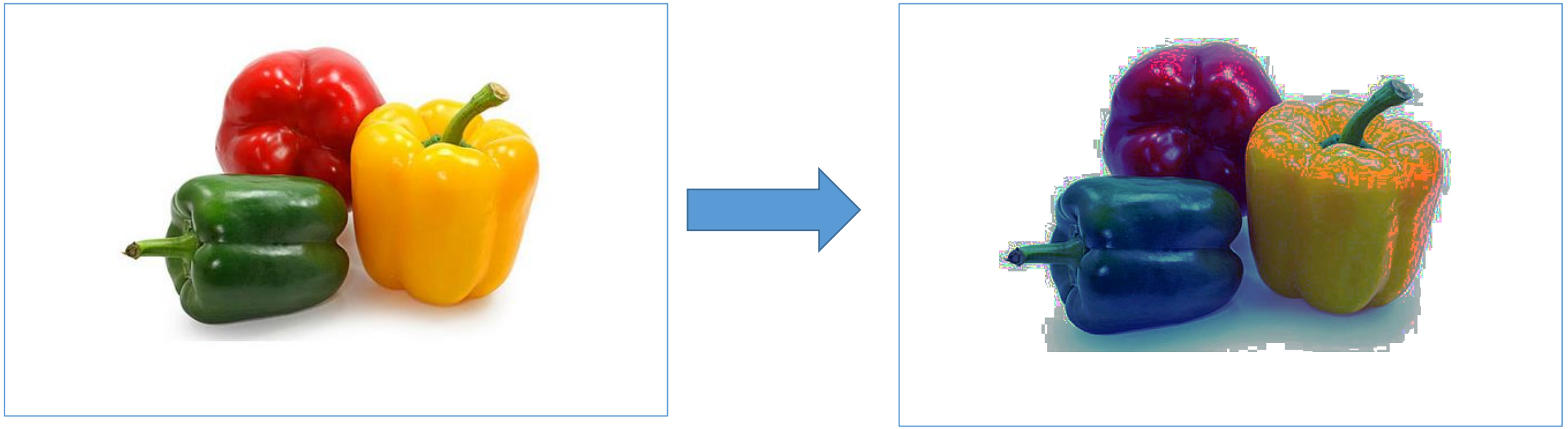
- The goal of contrast enhancement is normally altering intensities.
- Depending on project needs, Applying contrast enhancement to RGB matrices can generate undesired artifacts.

```
img=imread('peppers.jpg');
eqimg=zeros(size(img));
eqimg(:,:,1)=histeq(img(:,:,1));
eqimg(:,:,2)=histeq(img(:,:,2));
eqimg(:,:,3)=histeq(img(:,:,3));
eqimg=uint8(eqimg);
imshow(eqimg)
```

Source code for histogram equalization in the RGB color space

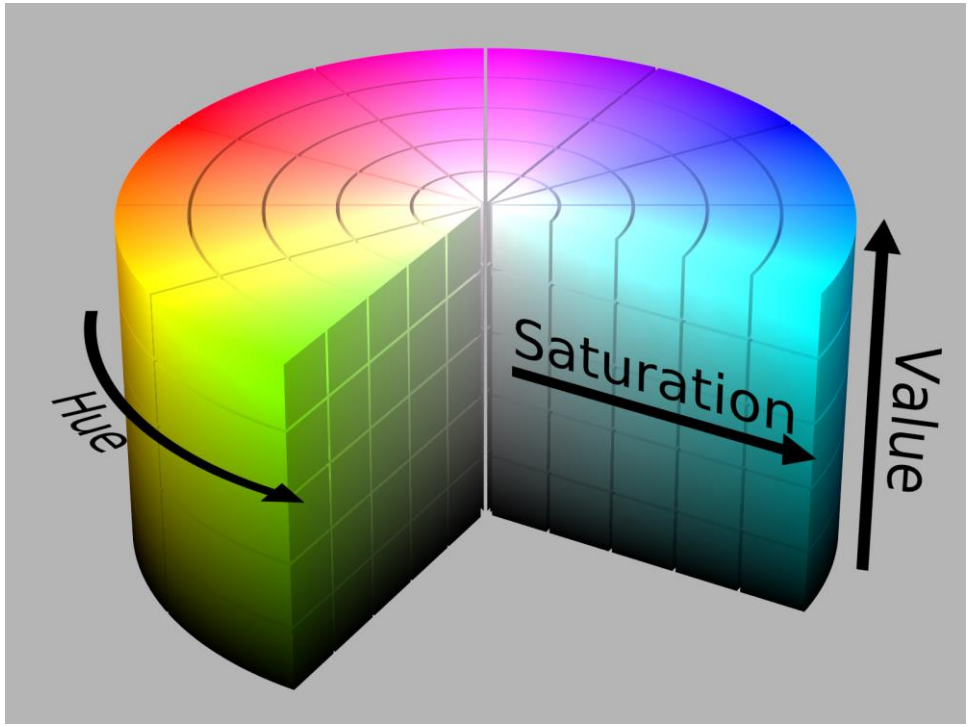
Histogram equalization on color images

- Using RGB equalizes colors instead of intensities. Results will not look natural



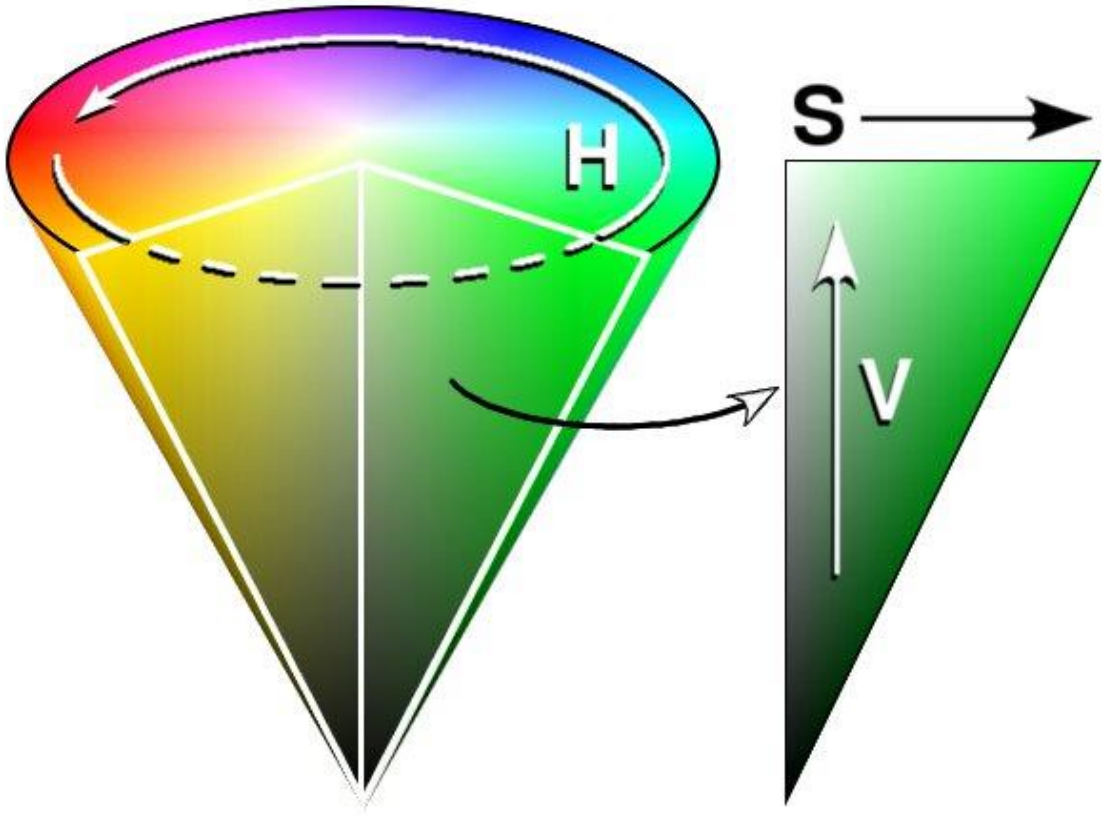
Histogram equalization on color images

- Solution: Use the HSV color space



Equalize the V matrix

Histogram equalization on color images



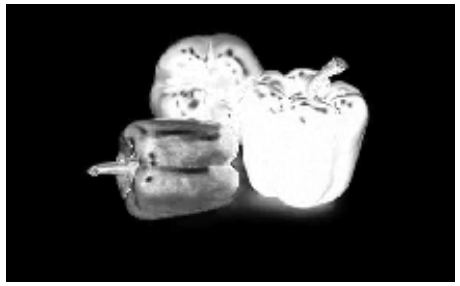
Equalize the V matrix

Histogram equalization on color images

Image

Hue

Saturation



value

```
img=imread('peppers.png');
imghsv=rgb2hsv(img);%Convert image to HSV space
subplot(2,2,1);
imshow(img);title('Image') %Original image
subplot(2,2,2);
imshow(imghsv(:,:,1));title('Hue') %Hue channel
subplot(2,2,3);
imshow(imghsv(:,:,2)); %Saturation channel
title('Saturation')
subplot(2,2,4);
imshow(imghsv(:,:,3));title('Value') %Value channel
```

Histogram equalization on color images

```
% Equalizing the histogram in the HSV domain
I=imread('autumn.tif'); % Read in image
Ihsv=rgb2hsv(I); % Convert original to HSV image
V=histeq(Ihsv(:,:,3)); % Histogram equalise V (3rd) channel of I
Ihsv(:,:,3)=V; % Copy equalised V plane into (3rd) channel I
Iout=hsv2rgb(Ihsv); % Convert I back to RGB form
subplot(1,2,1), imshow(I);
subplot(1,2,2), imshow(Iout);
```

Histogram equalization on color images

- Equalizing the bird image in the HSV color space

- For images with varying spatial characteristics its useful to apply local histogram equalization where the image is divided to sub-regions and histogram equalization is applied to each region independently.

Histogram matching

- Makes one image similar to another by matching their histograms

Source

Target

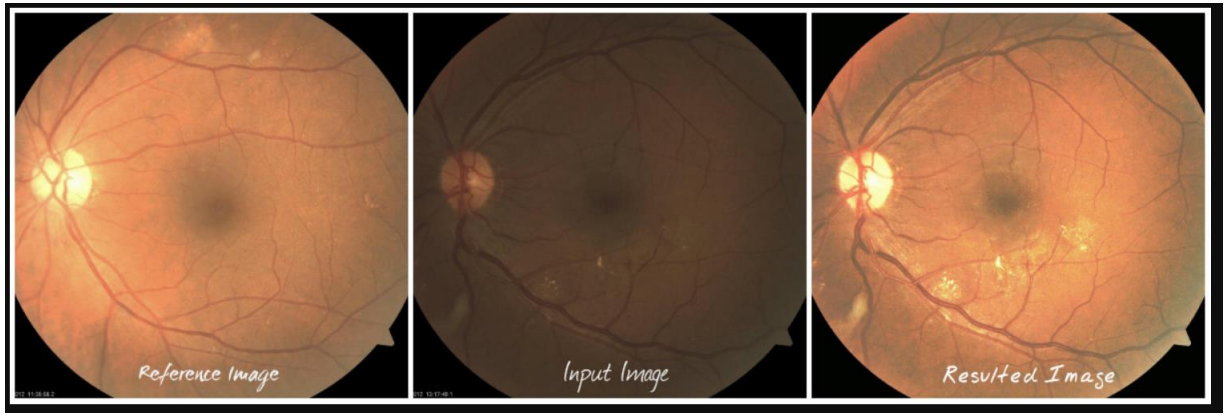
Matching Source to Target

Histogram matching

- Assume that two images are taken in two different lightening conditions and we like the histogram of the second image to closely match the histogram of the first image.

Histogram Matching

- Fluorescein angiography is a medical procedure in which a fluorescent dye is injected into the bloodstream The dye highlights the blood vessels in the back of the eye so they can be photographed. Examples of these images are shown below.
- The input image is taken in darker lightning condition. By matching the histogram of the second image with the reference image we get the image to the right.



https://studentathome.wordpress.com/2013/03/27/histogram-specification-matching/

Histogram Matching

- How to perform histogram matching between two images: Source S and Target T.
  1. Equalize the histogram of the source image S. You will get T_{Source} a transform that maps intensity values in *S*.
  2. Equalize the histogram of the desired image *T*. You will get T_{Target} a transform that maps intensity values in *T*.
  3. To map intensity values from S to T: Use T_{Source} to get the intensity values in the equalized image, next use T_{Target}^{-1} to get the intensity value matching D.

Histogram Matching

Source Image

0	1	3	4
1	2	2	3
1	3	4	4
3	2	5	2

Target Image

5	6	7	7
6	7	4	5
3	5	6	6
4	4	5	4

Histogram Equalization

Maximum possible intensity value

pdf value cdf value

Source image S

0	1	3	4
1	2	2	3
1	3	4	4
3	2	5	2

(a)

i	$\hat{h}_i$	$\hat{C}_i$	$7\hat{C}_i$
0	1/16	1/16	0
1	3/16	4/16	2
2	4/16	8/16	4
3	4/16	12/16	5
4	3/16	15/16	7
5	1/16	16/16	7
6	0/16	16/16	7
7	0/16	16/16	7

(b)

0	2	5	7
2	4	4	5
2	5	7	7
5	4	7	4

(c)

i	$\hat{h}_i$
0	1/16
1	0/16
2	3/16
3	0/16
4	4/16
5	4/16
6	0/16
7	4/16

(d)

Figure 5.9. Numerical example of histogram equalization: (a) a 3-bit image, (b) normalized histogram and CDF, (c) the equalized image, and (d) histogram of the result.

Reference: Kenny, The Art of Image Processing with Java,2010

Histogram Matching

0	1	3	4
1	2	2	3
1	3	4	4
3	2	5	2

Image *S*

i	T_{Source}
0	0
1	2
2	4
3	5
4	7
5	7
6	7
7	7

Image *S* equalization table T_{Source}

i	T_{Target}
0	0
1	0
2	0
3	0
4	2
5	4
6	6
7	7

Image *T* equalization table T_{Target}

5	6	7	7
6	7	4	5
3	5	6	6
4	4	5	4

Image *T*

Histogram Matching

2 Lookup its transform in T_{Source}

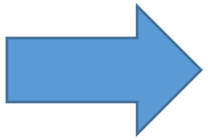
i	h_i
0	0
1	2
2	4
3	5
4	7
5	7
6	7
7	7

Image *S* equalization table T_{Source}

3 Use T_{Target}^{-1} to find correct match

i	h_i
0	0
1	0
2	0
3	0
4	2
5	4
6	6
7	7

Image *T* equalization table T_{Target}



1 Choose an output intensity

i	h_i
0	
1	
2	5
3	
4	
5	
6	
7	

Histogram matching table $T_{S \rightarrow T}$

Histogram Matching

2 Lookup its transform in T_{Source}

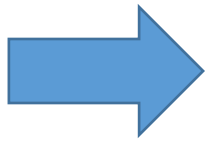
i	h_i
0	0
1	2
2	4
3	5
4	7
5	7
6	7
7	7

Image *S* equalization table T_{Source}

3 Use T_{Target}^{-1} to find correct match

i	h_i
0	0
1	0
2	0
3	0
4	2
5	4
6	6
7	7

Image *T* equalization table T_{Target}



1 Choose an output intensity

i	h_i
0	0
1	4
2	5
3	5
4	7
5	7
6	7
7	7

Can be either 5 or 6. Choosing 5 because it is closer to 3

Histogram matching table $T_{S \rightarrow T}$

Practical matters

- When displaying 8-bit grayscale images, color 0 maps to black and 255 to white.
- Its often that we like to display data that don't fall in this range but still like to cover the 256 possible values.
- One can perform histogram equalization, but a simpler method is the use of contrast stretching.
- In Matlab this is accomplished using imshow(img, []) or imagesc(img)
- These operations essentially apply the following equation

$$y = \frac{x - min\ intensity\ in\ the\ image}{\max intensity - \min intensity}$$

Exercise

- A grayscale image A was acquired in dark lightning conditions. Perform a histogram matching operation that transforms the intensity distribution of image A so that it matches image B where

$$\boldsymbol{A} = \begin{bmatrix} 135 & 23 & 80 \\ 49 & 129 & 23 \\ 65 & 65 & 129 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 251 & 129 & 124 \\ 109 & 249 & 124 \\ 110 & 255 & 255 \end{bmatrix}$$

- A and B use 8-bits to represent each pixel intensity value.

Answer: $A_{A->B} = \begin{bmatrix} 255 & 110 & 249 \\ 110 & 251 & 110 \\ 129 & 129 & 251 \end{bmatrix}$

Image filtering

- Image filtering: computes a function of local neighborhood at each position
- Its one of the most fundamental operations when dealing with image data or similar spatially varying data.
- Important for
  - Enhancing images

    Denoise, resize, increase contrast, etc.
  - Extracting information from images

    Texture, edges, distinctive points, etc.
  - Detecting patterns

    Template matching

Image filtering

- Two possible implementation of filters: Spatial domain and frequency domain.
- Spatial domain filters:
  - Direct manipulation of image pixels.
- Frequency domain filters:
  - Manipulation of Fourier transform coefficients or other frequency dividing transforms (*i.e.* Discrete Cosine Transform, Wavelets).

Image Filtering

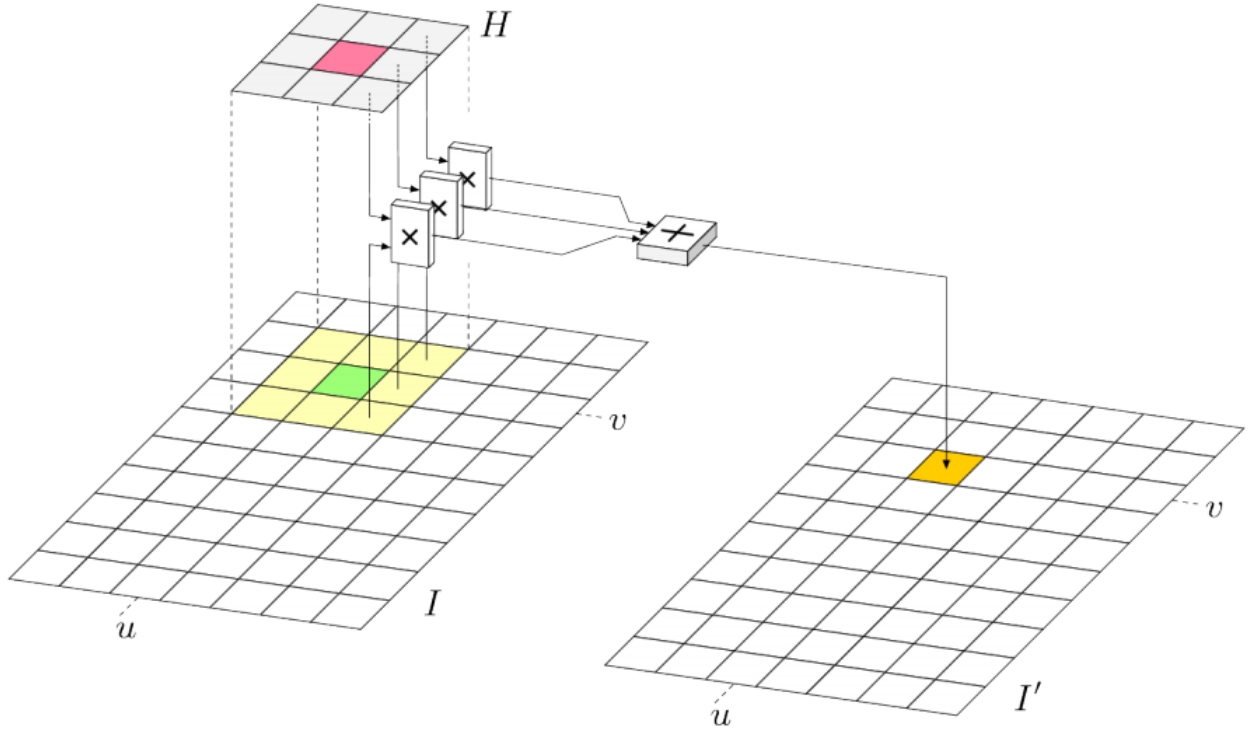
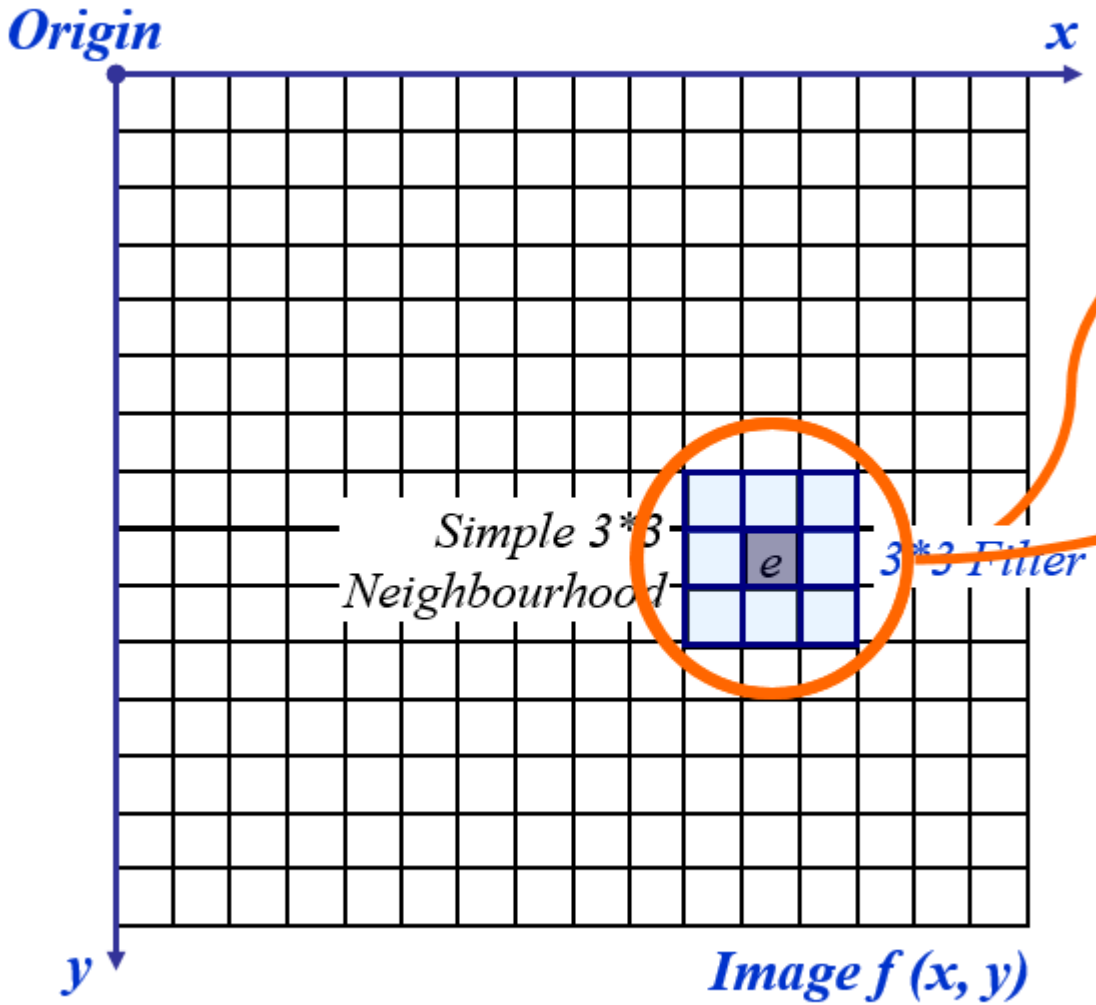


Figure: Linear filter. The filter matrix H is placed with its origin at position (u, v) on the image I. Each filter coefficient $H(i, j)$ is multiplied with the corresponding image pixel $I(u+i, v+j)$, the results are added, and the final sum is inserted as the new pixel value $I(u, v)$.

Image Filtering



a	b	c
d	e	f
g	h	i

Original Image Pixels

*

r	s	t
u	v	w
x	y	z

Filter

$$e_{processed} = v*e + r*a + s*b + t*c + u*d + w*f + x*g + y*h + z*i$$

Image Filtering

- Example of filters:

An averaging filter

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

a

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

b

- Apply filter (a) to the following matrix

$$M = \begin{bmatrix} 2 & 4 & 2 \\ 5 & 2 & 4 \\ 2 & 4 & 2 \end{bmatrix}, \; What\ is\ M(2,2)?$$

Image Filtering

- Apply filter (a) to the following matrix

$$M = \begin{bmatrix} 2 & 4 & 2 \\ 5 & 2 & 4 \\ 2 & 4 & 2 \end{bmatrix}, \; What\ is\ M(2,2)?$$

$$M(2,2) = \frac{2 \times 1 + 5 \times 1 + 2 \times 1 + 4 \times 1 + 2 \times 1 + 4 \times 1 + 2 \times 1 + 4 \times 1 + 2 \times 1}{9}$$

Image Filtering

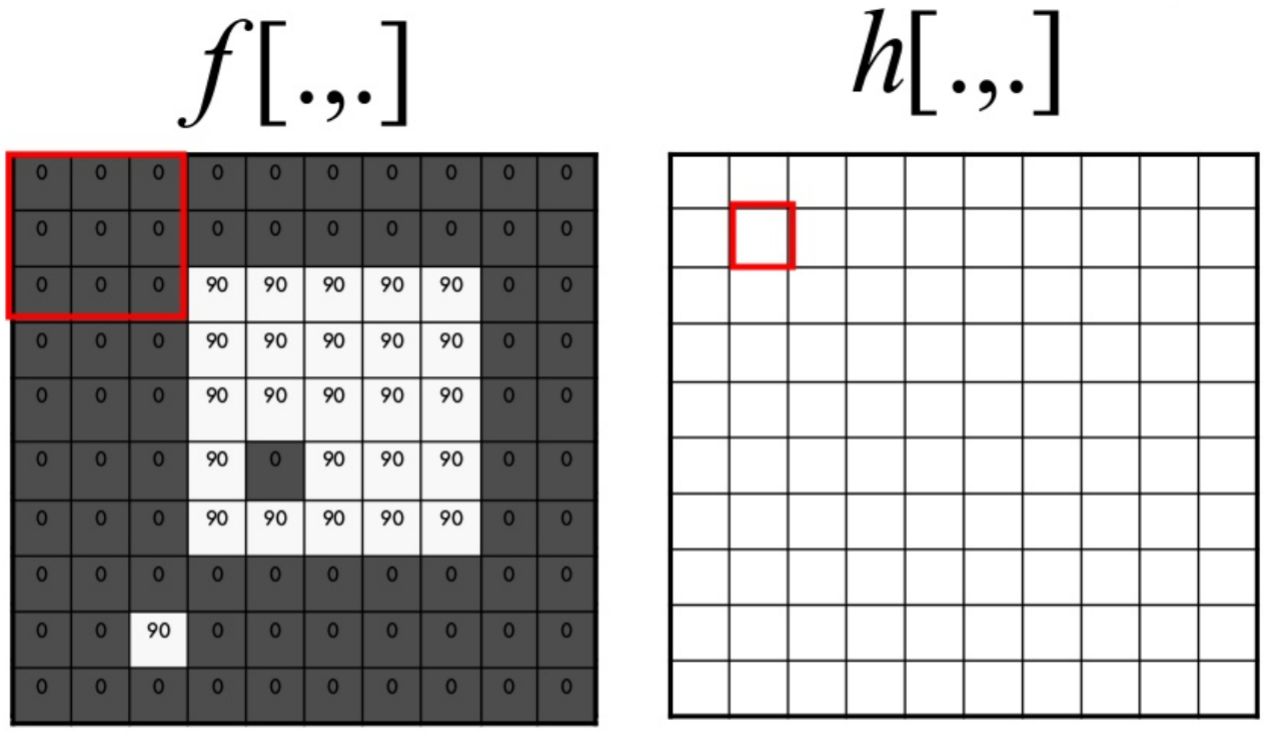
$$g[\cdot,\cdot]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Image Filtering

$g[\cdot,\cdot]$

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$



$$h[m,n]=\sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10							

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20						

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30					

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering

$g[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

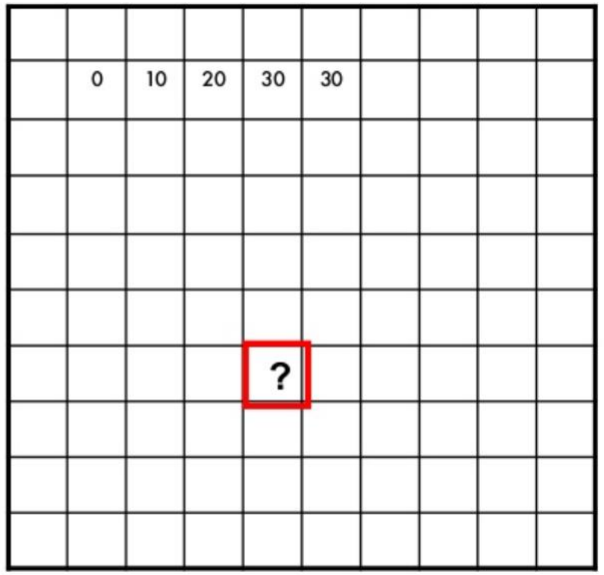
Image Filtering

$g[\cdot,\cdot]$

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[\cdot,\cdot]$

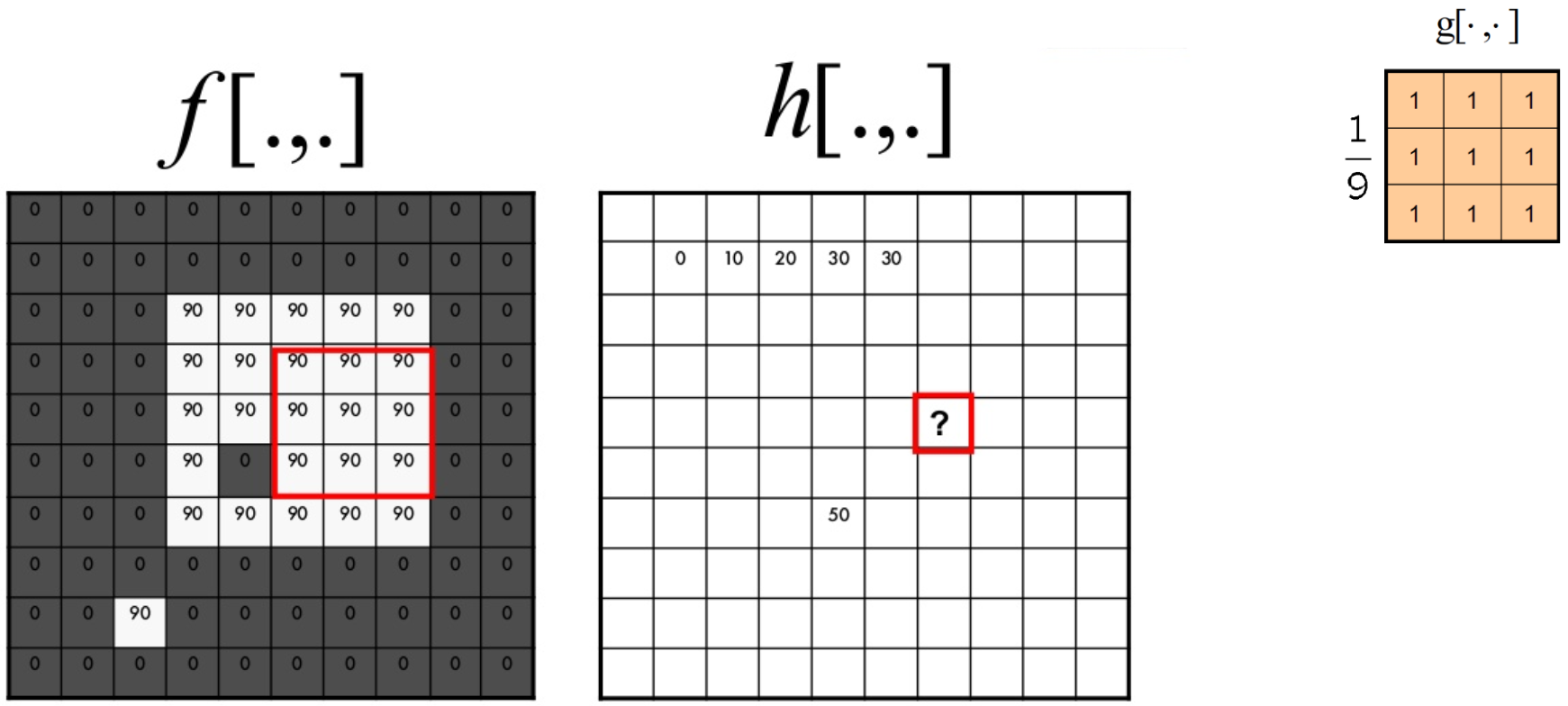
$h[\cdot,\cdot]$



$$h[m,n]=\sum_{k,l} g[k,l]\, f[m+k,n+l]$$

Credit: S. Seitz

Image Filtering



$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Credit: S. Seitz

Image Filtering

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Credit: S. Seitz

Image Filtering

Assume that this pixel value is noise. Applying the box filter will attenuate it

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Credit: S. Seitz

Image Filtering

Assume that this pixel value is noise. Applying the box filter will reduce its effect

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Credit: S. Seitz

What to do at the boundaries?

- Setting the boundary to a fixed intensity value.

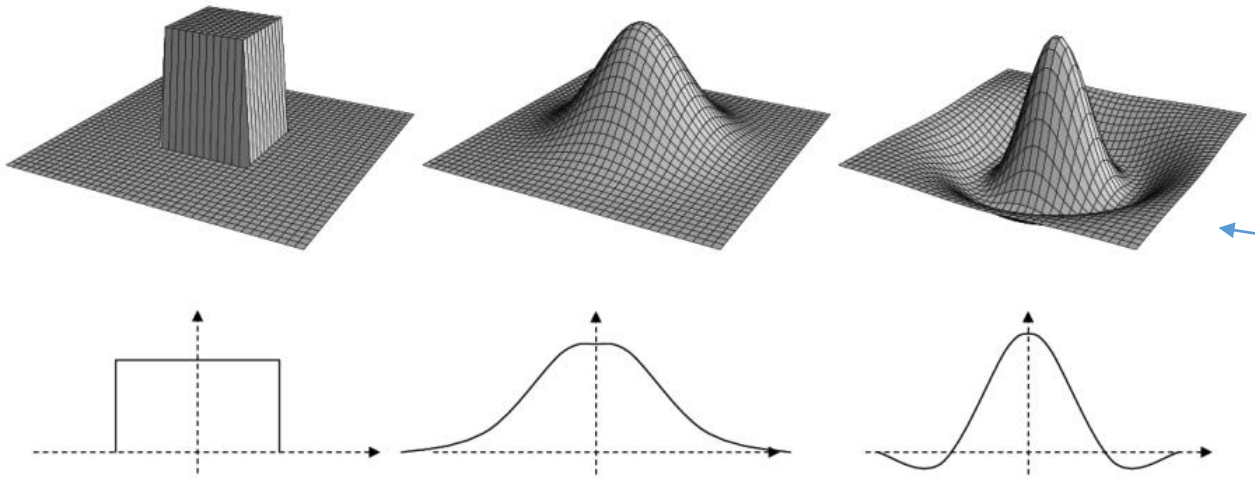
What to do at the boundaries

- Setting the boundary to a fixed intensity value.
- Mirroring: Mirror the last values at the boundary.

What to do at the boundaries

- Setting the boundary to a fixed intensity value.
- Mirroring: Mirror the last values at the boundary.
- Periodic extension: Boundary pixels takes intensity value equal to image pixels at the opposing edge.

Examples of filters



0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

(a)

0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

(b)

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

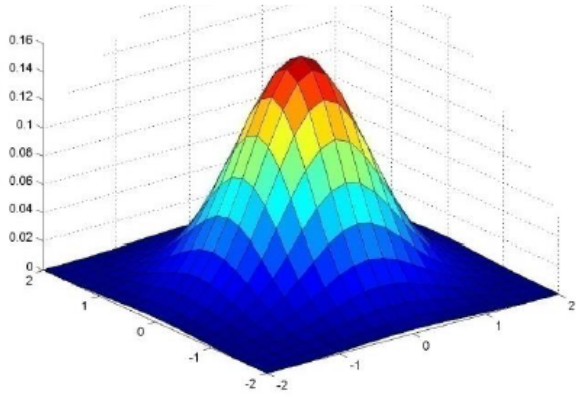
(c)

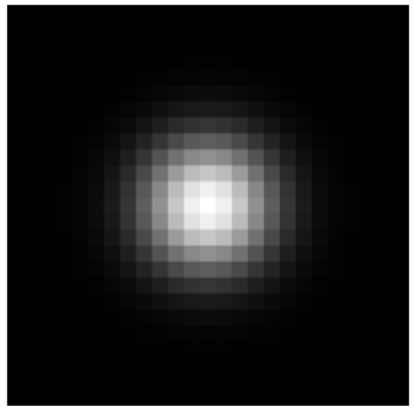
Used for edge detection

Figure: Filters a) Box Filter, b) Gaussian Filter, c) Laplace or Mexican Hat Filter

Gaussian filters

Weight contributions of neighboring pixels by nearness





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Like the box filter, the Gaussian is a smoothing filter.

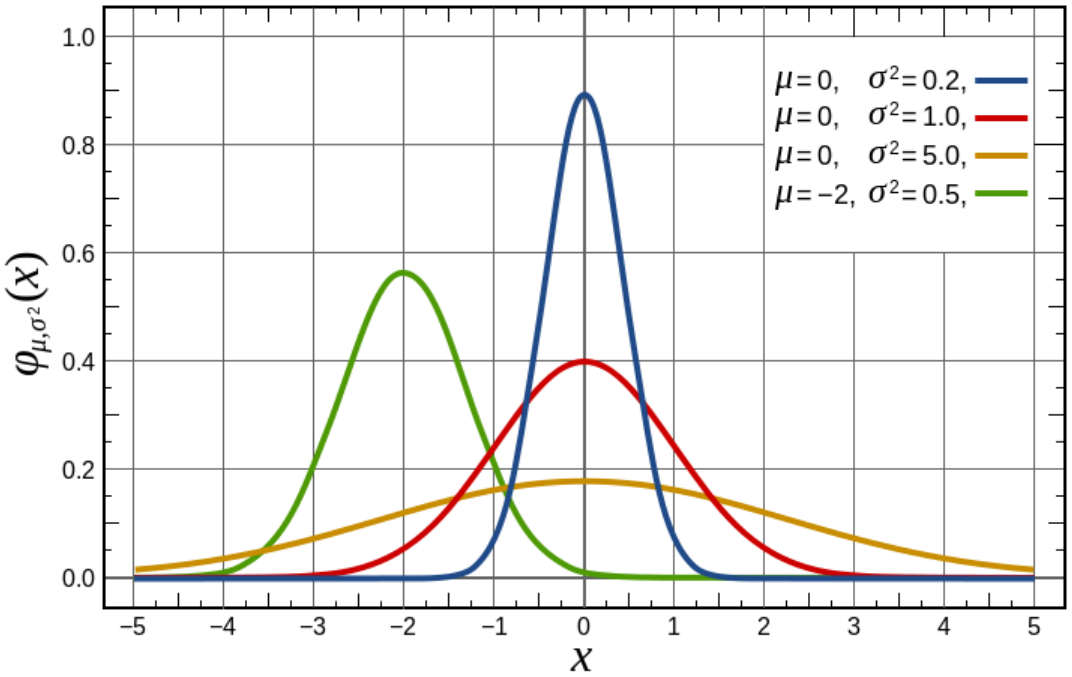
Use of filters for noise removal

- Filter are typically used to clean images from noise.
- Examples of noise artifacts: Gaussian noise, impulsive noise, and ‘salt and pepper’.

Gaussian noise

- Additive and normally distributed

$$f(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



- Generate Gaussian normal values (μ=0, δ=1) using ‘randn’.

```
>> r = 3 + 2.*randn(3,3)

r =

    5.8181    0.5850    3.9778
    5.8344    4.4345    5.0694
    4.3430    6.2605    4.4538
```

Random numbers with a mean of 3 a σ value of 2

Gaussian noise

σ=10

σ = 25

Linear filtering

- imfilter.m

```
imfilter N-D filtering of multidimensional images.
   B = imfilter(A,H) filters the multidimensional array A with the
   multidimensional filter H.  A can be logical or it can be a
   nonsparse numeric array of any class and dimension.  The result,
   B, has the same size and class as A.

   Each element of the output, B, is computed using double-precision
   floating point.  If A is an integer or logical array, then output
   elements that exceed the range of the given type are truncated,
   and fractional values are rounded.

   B = imfilter(A,H,OPTION1,OPTION2,...) performs multidimensional
   filtering according to the specified options.  Option arguments can
   have the following values:

   - Boundary options

       X             Input array values outside the bounds of the array
                     are implicitly assumed to have the value X.  When n
                     boundary option is specified, imfilter uses X = 0.

       'symmetric'   Input array values outside the bounds of the array
                     are computed by mirror-reflecting the array across
                     the array border.

       'replicate'   Input array values outside the bounds of the array
```

Removing Gaussian noise using the box and Gaussian filters

- Size of the filter in both cases is 3x3.

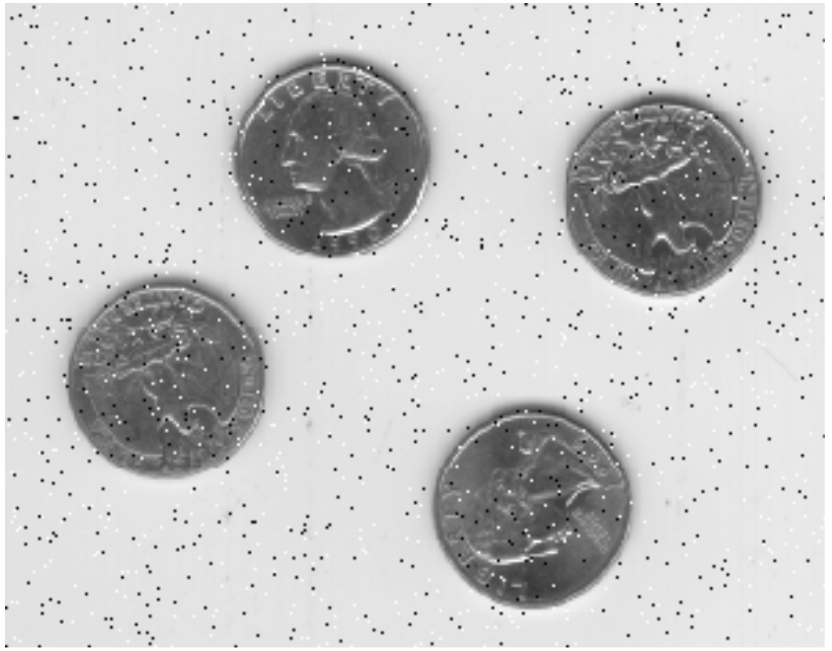
σ=25

Box filtering

Gaussian filtering

Salt and Pepper Noise

- Caused by transmission error. Corrupted pixels are assigned the value of 0 or the maximum possible value.



Median filtering: Why does it work

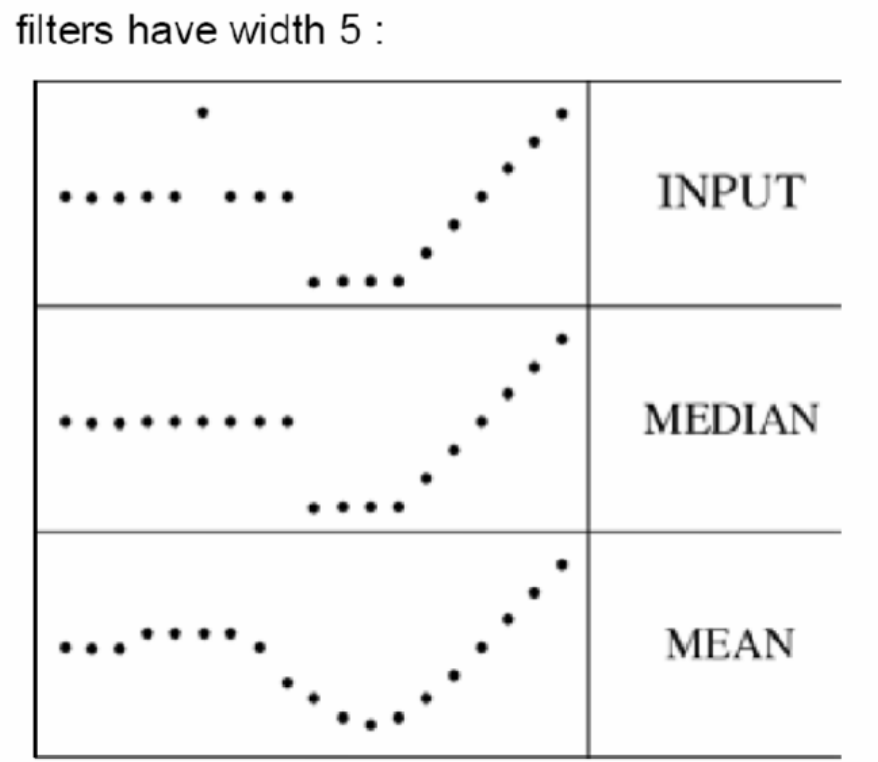
$$M = \begin{bmatrix} 126 & 135 & 136 \\ 128 & 0 & 136 \\ 122 & 130 & 134 \end{bmatrix}$$

- S&P noise affects M(2,2).
- Using median filtering M(2,2) = median (126,135, 136, 128, 0, 136, 122, 130, 134).
- Sorting intensity values: 0, 122,126,128,130,134,135,136,136.

  Median = mid-value = 130.
- S&P values are outliers and taking the median value computes new pixel intensities while ignoring S&P effect.

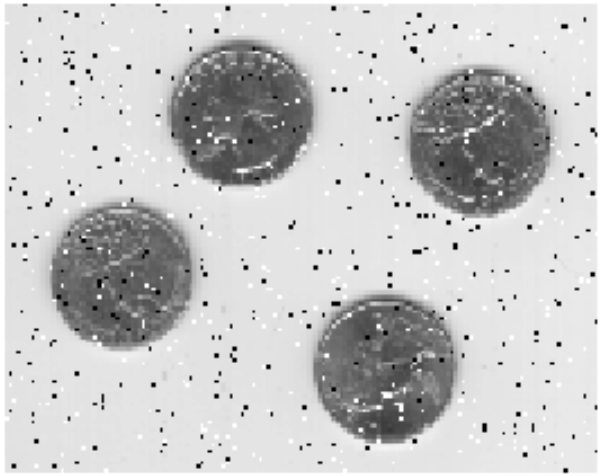
Median Filter

- What advantage does median filtering have over Gaussian filtering?
  - Robustness to outliers



Median Filter

(a)



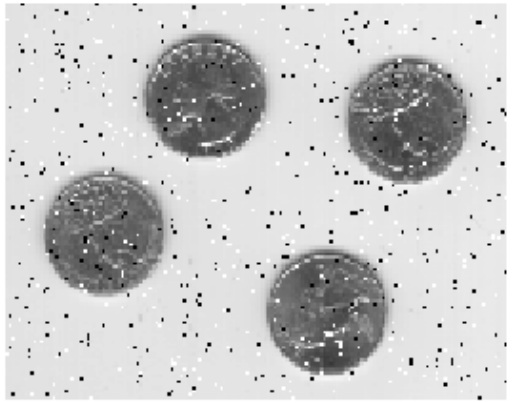
(b)

(c)

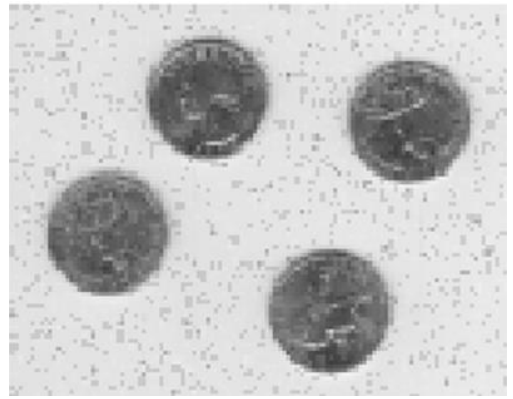
(a) original image, (b) image+ ‘salt and pepper’ noise (c) After median filtering

Comparison with a Gaussian

(a)



(b)



(c)

(a) original image, (b) image+ ‘salt and pepper’ noise (c) After Gaussian filtering