Week 8 – Robotic Vision 1

Advanced Robotic Systems – MANU2453

Dr Ehsan Asadi, School of Engineering
RMIT University, Victoria, Australia
Email: ehsan.asadi@rmit.edu.au

Lectures

Wk	Date	Lecture (NOTE: video recording)	Maths Difficulty	Hands-on Activity	Related Assessment
1	24/7	• Introduction to the Course • Spatial Descriptions & Transformations			
2	31/7	• Spatial Descriptions & Transformations • Robot Cell Design			Robot Cell Design Assignment
3	7/8	• Forward Kinematics • Inverse Kinematics			
4	14/8	• ABB Robot Programming via Teaching Pendant • ABB RobotStudio Offline Programming		ABB RobotStudio Offline Programming	Offline Programming Assignment
5	21/8	• Jacobians: Velocities and Static Forces			
6	28/8	• Manipulator Dynamics			
7	11/9	• Manipulator Dynamics		MATLAB Simulink Simulation	
8	18/9	• Robotic Vision		MATLAB Simulation	Robotic Vision Assignment
9	25/9	• Robotic Vision		MATLAB Simulation	
10	2/10	• Trajectory Generation			
11	9/10	• Linear & Nonlinear Control		MATLAB Simulink Simulation	
12	16/10	• Introduction to I4.0 • Revision			Final Exam

Content

- Introduction to Robotic Vision & Image Processing
- The Digital Image
- Pixel Point Processing
- Pixel Group Processing
- Geometric Transformation
- Feature Extraction

Content

- Introduction to Robotic Vision & Image Processing
- The Digital Image
- Pixel Point Processing
- Pixel Group Processing
- Geometric Transformation
- Feature Extraction

Introduction to Robotic Vision

- Robots are goal-oriented machines that can sense, plan and act.

Where are objects? Vision

Where am I?

encoders

GPS

How do I get there (e.g. to catch the rabbit)?

https://commons.wikimedia.org/wiki/File:Cartoon_Robot.svg

Mobile Robot

Introduction to Robotic Vision

- One solution would be to use GPS.

https://pixabay.com/en/gps-map-maps-navigation-direction-3108704/

- However, it has some disadvantages:
  - Satellites are obscured in urban areas, mining areas, in-door, underwater, underground, deep forest.
  - Signal bounces off metal object.
  - Water and dew affects the accuracy.
- Also, GPS only tells me where I am, but not where the object is, unless we attach a GPS device also on the target (the rabbit).

Introduction to Robotic Vision

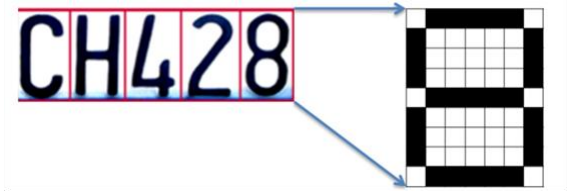
- Animals and humans use sensors to accomplish the tasks!
- Vision is perhaps the most important of all the sensors:
- Eyes are essential for:
  - Finding food;
  - Avoid being hunted;
  - Finding mates etc.
- Eyes can sense shape, colour, motion…
- Eyes are long range sensor!
  - Beyond the finger tip.

https://en.wikipedia.org/wiki/Eagle_eye

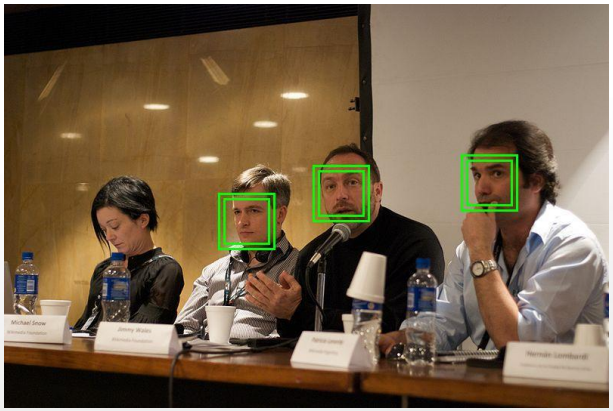
- By seeing the surrounding, animals and humans will be able to know where they are, e.g. currently under a tree.
- And, they will know where the target is because they can see the target.

Introduction to Robotic Vision

- Robotic vision allows robots to achieve the previous tasks.
- Apart from these, robotic vision are also used in many other tasks:
  - Font recognition
    e.g. to detect car plate
  - Face recognition
  - Iris recognition
    e.g. for login w/o password
  - Fingerprint recognition
  - Object recognition
    e.g. self-driving car
    recognise pedestrians,
    other cars, lines etc.
  - Google search by image
  - Etc. Etc.



http://www.docs4ecm.com/ocr.html



https://commons.wikimedia.org/wiki/File:Face_detection.jpg

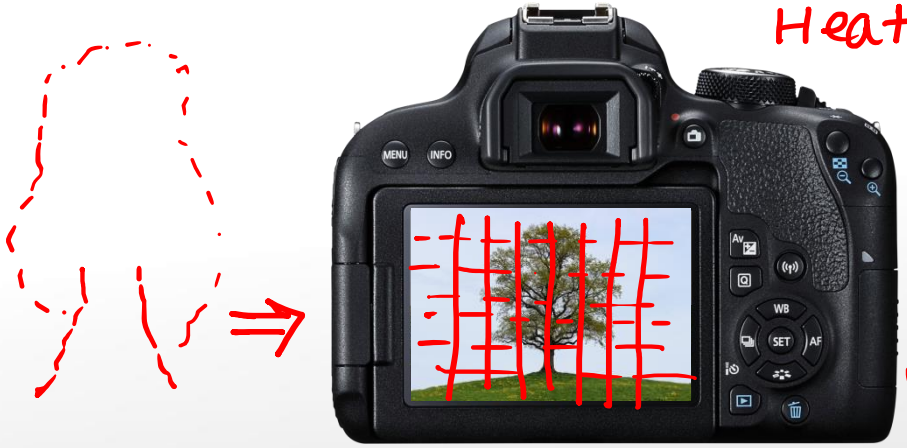
Introduction to Robotic Vision

- For industrial robotic manipulators, robotic vision is also used for the following tasks:
  - Recognize the object (e.g. shape, colour etc.)
  - Determine the position and orientation of the object with respect to the end-effector.
  - Quality control.

https://www.flickr.com/photos/departmentofenergy/11343380734

Introduction to Image Processing

- To implement robotic vision, we need some cameras to take pictures:
  - Colour images
  - Grey scale images
  - Black and white images
  - Other types are possible for e.g. Infrared Camera, 3D Camera, etc.



- The pictorial information is a two-dimensional visual image.

Introduction to Image Processing

- Image processing then refers to the manipulation and analysis of pictorial information to:
  - Improve an image
  - Correct an image
  - Analyse an image
  - Change an image

Original

Sharpened

Threshold

Sobel Combined Edge Detection

Content

- Introduction to Robotic Vision & Image Processing
- The Digital Image
- Pixel Point Processing
- Pixel Group Processing
- Geometric Transformation
- Feature Extraction

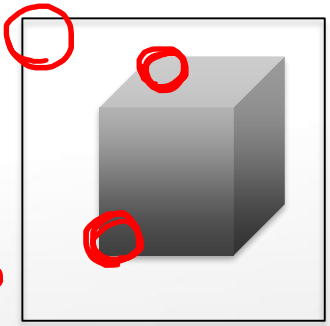
The Digital Image

- A real-world image varies continuously in shades and colours.
- We will focus on grey-scale images first.
- When we use a digital camera to take the picture, the continuous image is divided into individual points of brightness.
- This is followed by describing each point of brightness using a digital data value.
  - 255 for pure white, 0 for pure black.

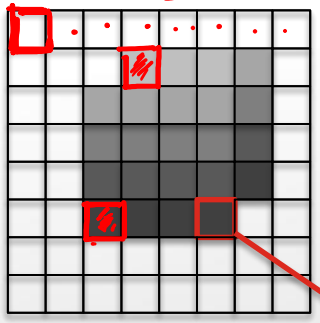
8 bits unsigned Integer (2^8) 256 0,..,255

CCD

Digital cam



Continuous Tone Image



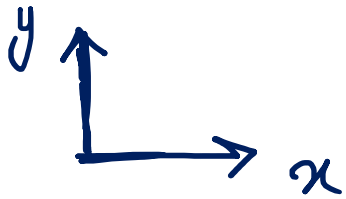
Sampled Image

Pixel

255	255	255	255	255	255	255	255
255	255	255	130	130	130	120	255
255	255	120	120	120	120	110	255
255	255	110	110	110	110	100	255
…							
		20					

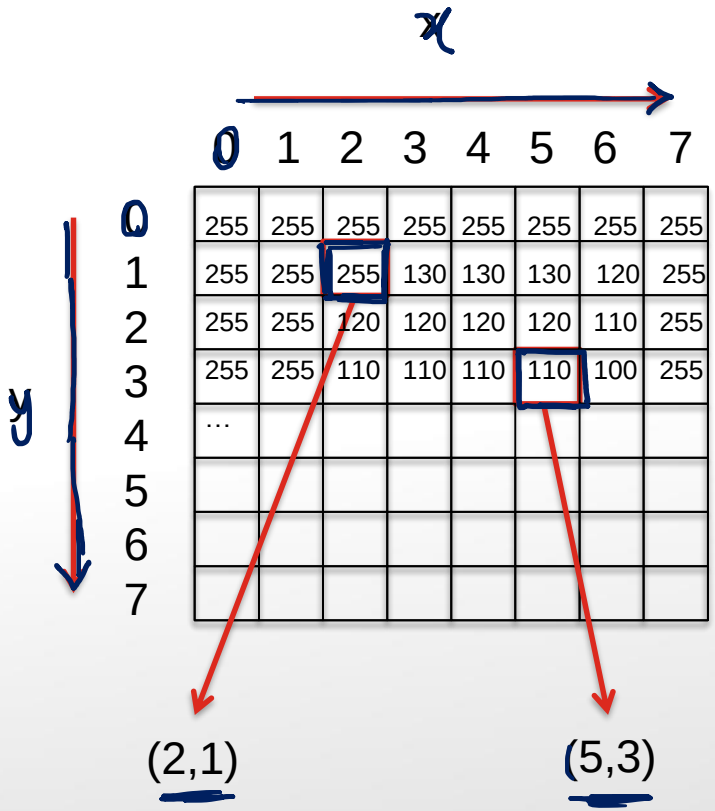
Quantized Values

The Digital Image



- An image is generally sampled into a rectangular array of pixels.
- Each pixel has an (x, y) coordinate:
  - x from left to right
  - y from top to bottom!!!
  - This is useful for computer programs as they display values from top to bottom.
- The brightness value for a pixel is written as $I(x, y)$. E.g.
  - $I(2,1) = 255$
  - $I(5,3) = 110$

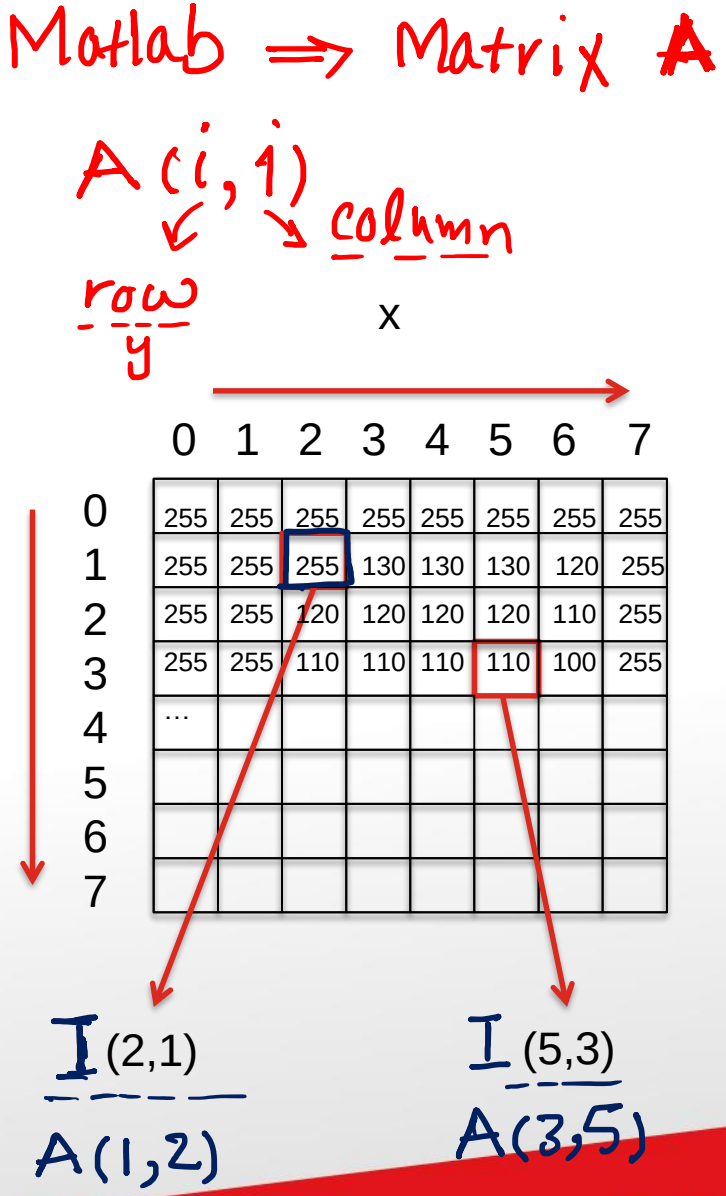
$I(x,y) =$



The Digital Image

- Very important:
- Matlab's row and column convention is different from our usual understanding of "coordinates".
- For e.g. if we call the matrix on the right as "A";
- Then A(1,2) is in our usual coordinates understanding actually the $(2,1)^{th}$ component.
- Similarly, A(3,5) is the $(5,3)^{th}$ component.
- To summarize: A(i,j)

row column



MATLAB Codes

- MATLAB Code for Importing Image:

Matrix

```
I = imread('cameraman.tif');
```

- This will generate an array of intensity values:

Editor - Vision_Cameraman_Grey.m

Variables - I

I

256x256 double

	1	2	3	4	5	6	7	8	9	10	11
1	156	159	158	155	158	156	159	158	157	158	158
2	160	154	157	158	157	159	158	158	158	160	155
3	156	159	158	155	158	156	159	158	157	158	158
4	160	154	157	158	157	159	158	158	158	160	155
5	156	153	155	159	159	155	156	155	155	157	155
6	155	155	155	157	156	159	152	158	156	158	152
7	156	153	157	156	153	155	154	155	157	156	155
8	159	159	156	158	156	159	157	161	162	157	157
9	158	155	158	154	156	160	162	155	159	161	156
10	155	154	157	158	160	160	159	160	158	161	160
11	154	157	157	157	156	155	159	154	159	158	161
12	152	150	155	154	152	156	157	156	157	154	157
13	157	153	156	155	157	160	160	157	159	159	160
14	151	154	157	156	156	158	158	156	157	159	158
15	156	157	157	160	159	159	156	158	159	162	161

MATLAB Codes

- To show the image:

```
figure, imshow(I)
```

- The figure will be shown.

Image Quality

- The quality of the digital image (i.e. how well the digital image represents the original scene) is determined by:
  - Spatial Resolution (Num of pixels / size)
  - Brightness Resolution (Brightness / intensty)
- Spatial resolution is determined by the number of pixels in the image:

259 x 194

52 x 39

26 x 20

https://www.pexels.com/photo/greyscale-photo-of-bench-beside-tall-building-999334/

Image Quality

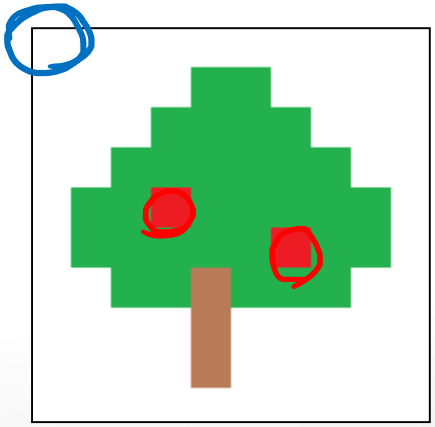
- Brightness Resolution refers to how accurately the digital pixel's brightness can represent the intensity of the original image.
- We already mentioned that 0 represents black and 255 represents white. But this is only true for "8-bit" grey scale. (8 bit = 2^8 = 256 levels).

- Some devices only have "3-bit" grey scale, thus only 2^3 = 8 levels. 0 represents black and 7 represents white.

- Or.. "4-bit" grey scale, where 0 represents black and 15 represents white.
- Or.. "5-bit" grey scale, where 0 represents black and 31 represents white.
- Etc.

Colour Images

- So far we have discussed about grey scale images.
- What about colour images?
- Concept of sampling and quantization still apply.
- However, we now have three "layers" for the quantized values – one for red, green and blue each.



Colour Image

Red

255	255	255	255	255	255	255	255	255	255
255	255	255	255	34	34	255	255	255	255
255	255	255	34	34	34	34	255	255	255
255	255	34	34	34	34	34	34	255	255
255	34	34	237	34	34	34	34	34	255
255	34	34	34	34	34	237	34	34	255
255	255	34	34	185	34	34	34	255	255
255	255	255	255	185	255	255	255	255	255
255	255	255	255	185	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255

Green

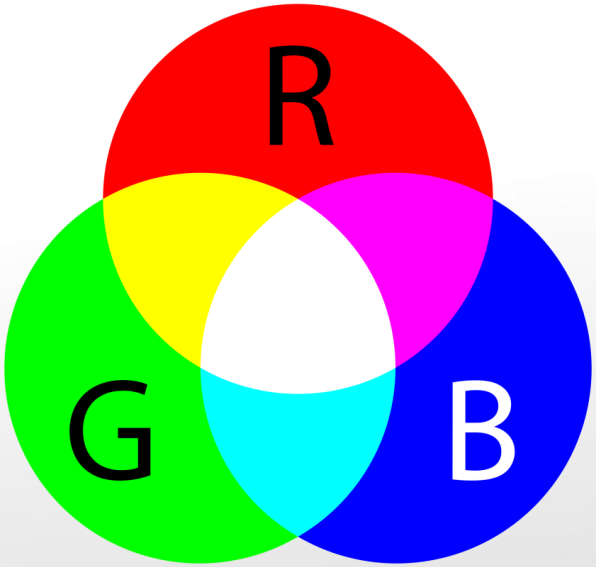
255	255	255	255	255	255	255	255	255	255
255	255	255	255	177	177	255	255	255	255
255	255	255	177	177	177	177	255	255	255
255	255	177	177	177	177	177	177	255	255
255	177	177	28	177	177	177	177	177	255
255	177	177	177	177	177	28	177	177	255
255	255	177	177	122	177	177	177	255	255
255	255	255	255	122	255	255	255	255	255
255	255	255	255	122	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255

Blue

255	255	255	255	255	255	255	255	255	255
255	255	255	255	76	76	255	255	255	255
255	255	255	76	76	76	76	255	255	255
255	255	76	76	76	76	76	76	255	255
255	76	76	36	76	76	76	76	76	255
255	76	76	76	76	76	36	76	76	255
255	255	76	76	87	76	76	76	255	255
255	255	255	255	87	255	255	255	255	255
255	255	255	255	87	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255

Colour Images

- The three colours (Red, Green and Blue) mix together to create all other colours.
  - E.g. Brown = 185 red, 122 green, 87 blue.
  - E.g. White = 255 red, 255 green, 255 blue.
- This is called additive colour property.



https://en.wikipedia.org/wiki/RGB_color_model

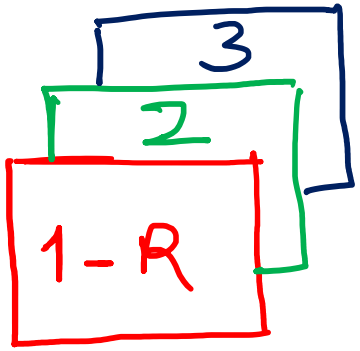
Colour Images

- If we want to convert a colour image into grey scale image, simply calculate the average of the three colour intensities:

$$I(\text{grey}) = \frac{I(\text{red}) + I(\text{green}) + I(\text{blue})}{3}$$

Colour Red Green Blue Grey

MATLAB Codes

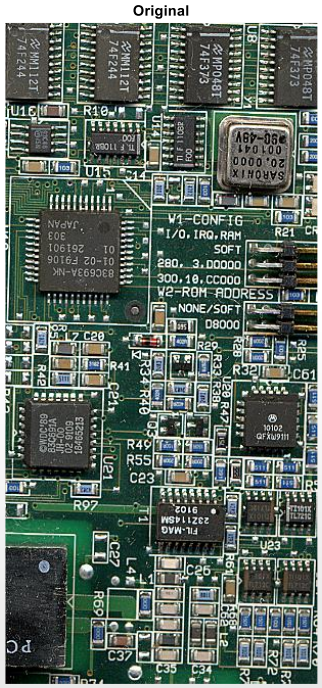


- Import a colour image:

```
I = imread('board.tif');
```

- Show the colour image:

```
figure, imshow(I)
```



I(:,:,1)

row

Column

z

I(:,:,2)

I(:,:,3)

MATLAB Codes

uint8(255) → double

- To convert into grey scale image:

```
IRed = double(I(:,:,1));
IGreen = double(I(:,:,2));
IBlue = double(I(:,:,3));
IGrey = (IRed+IGreen+IBlue)/3;
I = uint8(IGrey);
```

Convert 8-bit integer to double. Otherwise, when we add the numbers, they may get saturated at 255.

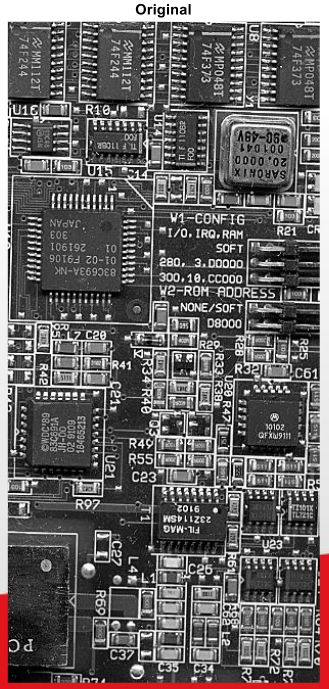
Convert back to 8-bit integer, because "imshow" requires 8-bit integer.

- Show the grey scale image:

```
figure, imshow(I)
```

uint8

255
+255
+255 > 255

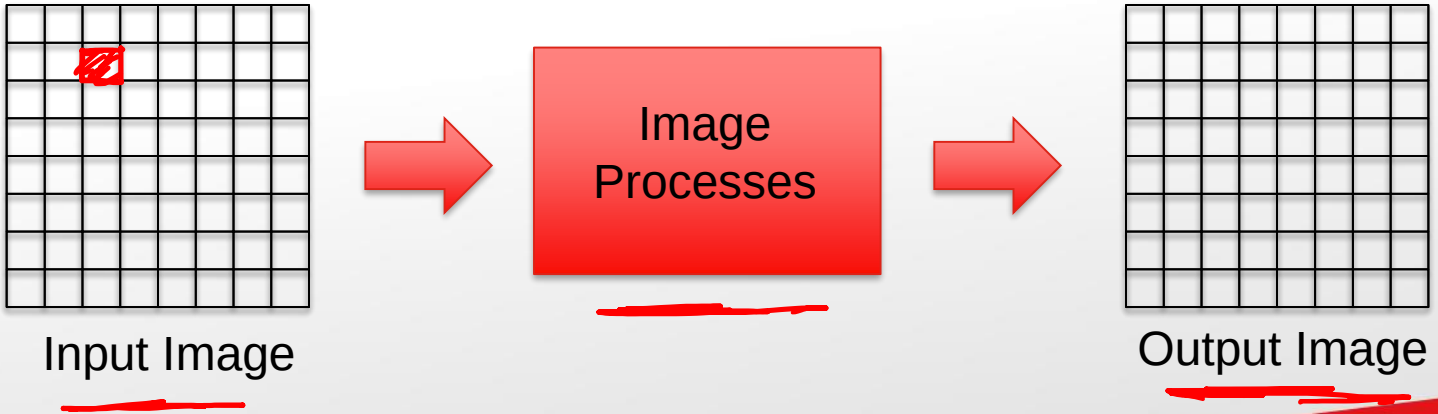


Content

- Introduction to Robotic Vision & Image Processing
- The Digital Image
- Pixel Point Processing
- Pixel Group Processing
- Geometric Transformation
- Feature Extraction

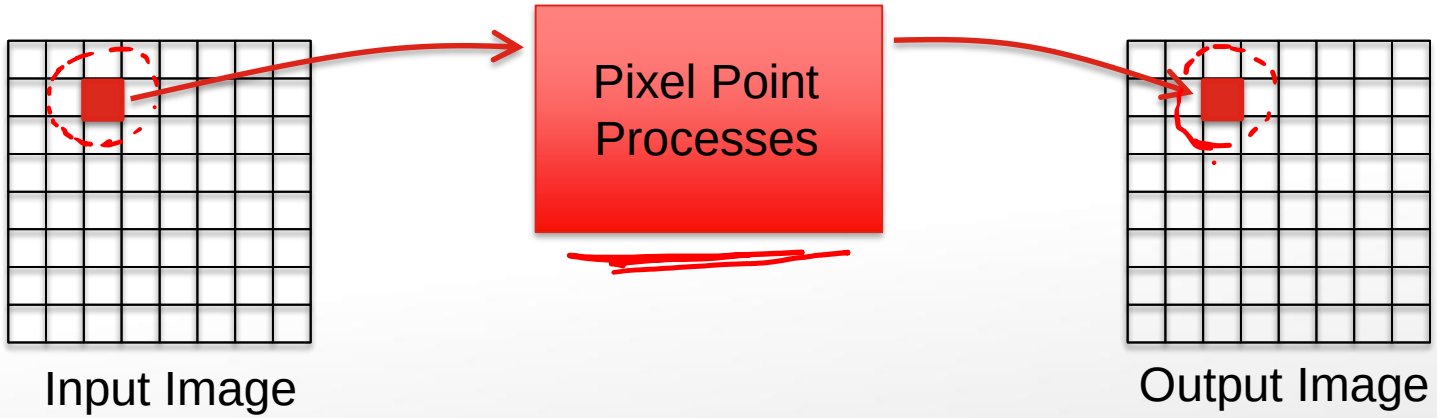
Image Processing

- We already mentioned that image processing means the manipulation and analysis of pictorial information (in this case pixel values) to:
  - Improve an image
  - Correct an image
  - Analyse an image
  - Change an image
- Let's differentiate between an input and an output image:



Pixel Point Processing

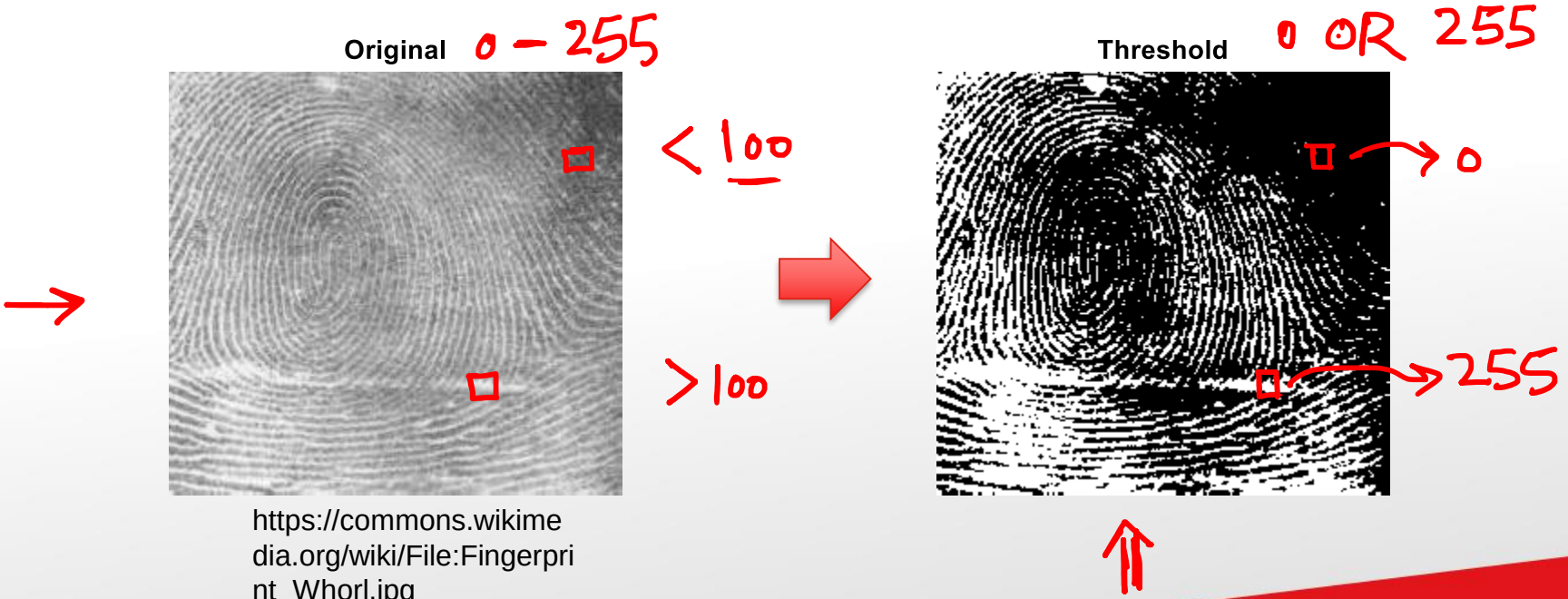
- The simplest of these would be "Pixel Point Processing".
- The grey level of each pixel in the input image is modified to a new value (often by mathematical or logical relationship), then…
- … placed in the output image at the same spatial location.



Thresholding

Value / Threshold

- One example of pixel point processing would be the "thresholding" process.
  - Pixels of brightness less than a threshold will be set to black (0);
  - Pixels of brightness higher than the threshold will be set to white (255).
- Useful for enhancing contrast:



https://commons.wikimedia.org/wiki/File:Fingerprint_Whorl.jpg

Thresholding

- Also useful for highlighting only “important” information.
  - E.g. The cameraman stands out from the background.

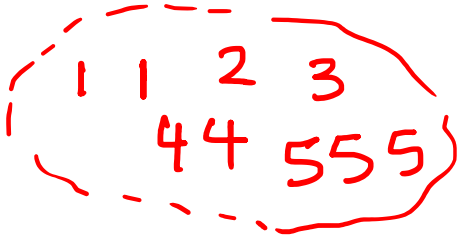
remove

Original

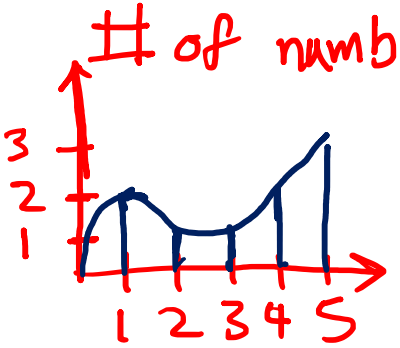
Threshold

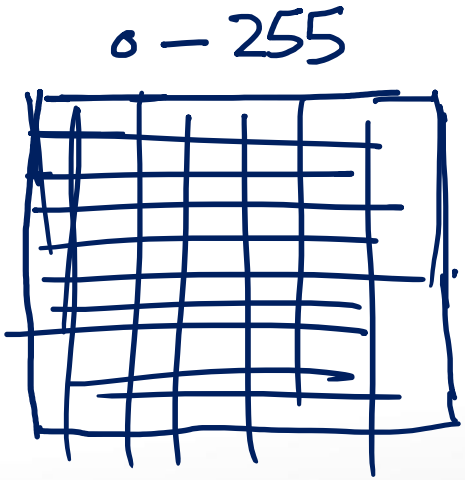
what is a good Threshold ?

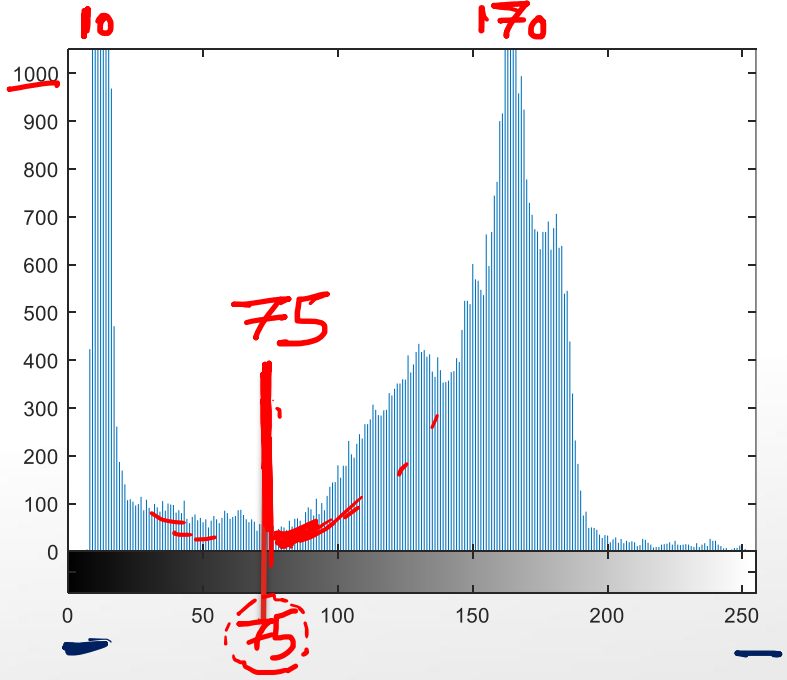
Brightness Histogram



- But what is a good "threshold" for the previous technique?
- We can use the histogram of brightness values as a guide.
  - This is the histogram for the Cameraman example:







- In this example, it may be a good idea to set any pixel values less than 75 as 0, and higher than 75 as 255.

MATLAB Codes

- The codes to get the histogram, and perform the pixel-by-pixel thresholding are as follows:

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

```
figure, imhist(I)
```

Show histogram

```
I=double(I);
```

```
[m,n]=size(I);
```

Get the image size

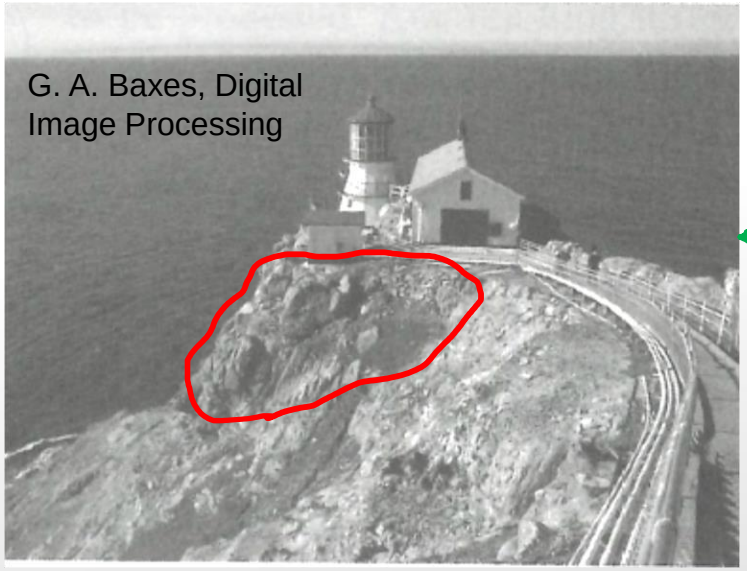
```
Ithreshold = zeros(m,n);

for i = 1:m
    for j = 1:n
        if I(i,j) > 75
            Ithreshold(i,j) = 255;
        else
            Ithreshold(i,j) = 0;
        end
    end
end

Ithreshold = uint8(Ithreshold);
figure,imshow(Ithreshold)
title('Threshold')
```

To adjust

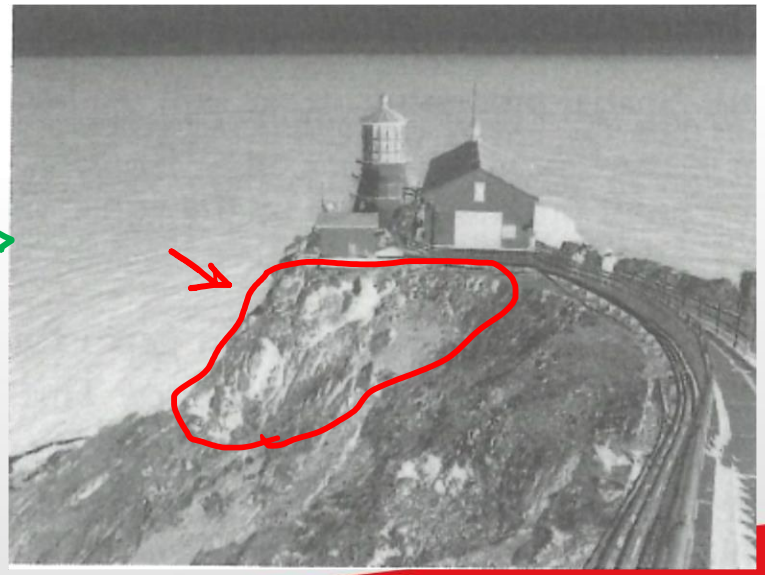
Thresholding

Complement Image Operation

- Another pixel point process is making a negative image from a positive one.
- Black (0) maps to white (255) and vice versa.
- This is sometimes useful, as the human eye is more sensitive to slight brightness changes in dark regions of an image than in light regions.
  - The undetectably subtle brightness shades in input image's bright areas become clearly visible after transforming to dark areas in output image.

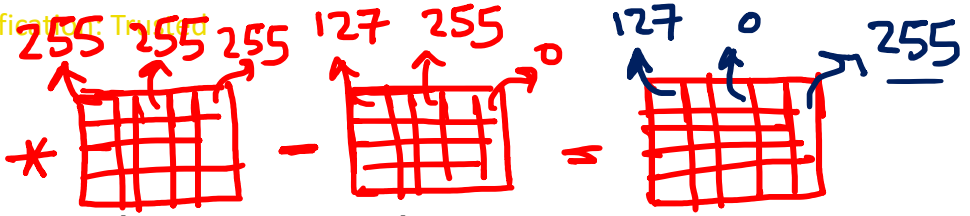


Original



Complement

MATLAB Codes



- The MATLAB Codes for the complement image operation are:

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

```
I=double(I);
```

```
[m,n]=size(I);
```

Get the image size

```
Inegative = ones(m,n)*255-I;

Inegative = uint8(Inegative);
figure,imshow(Inegative)
title('Negative')
```

"ones(m,n)" creates a matrix of size m x n, with all entries as 1. Multiply this with 255, then subtract the original value.

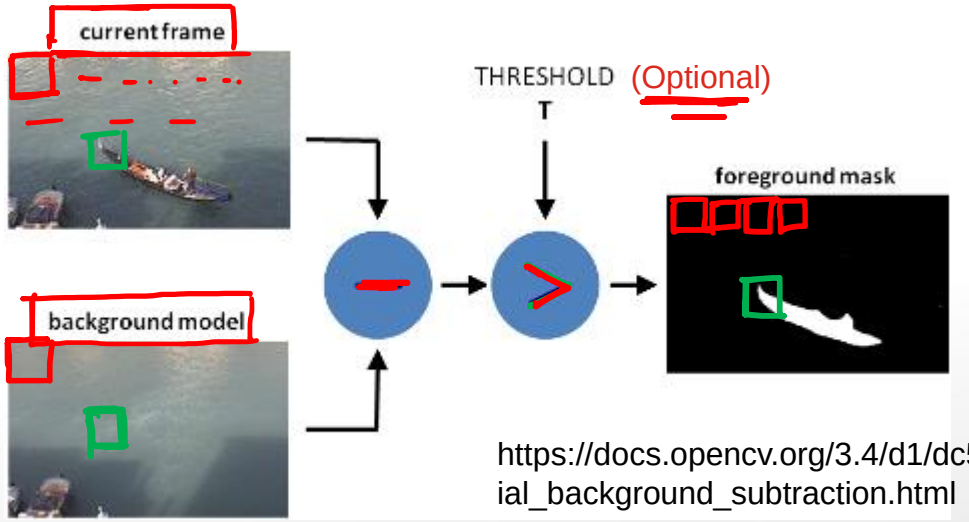
Negative

Differencing

- Differencing is also a pixel point operation, but it requires multiple images.

Output = Input1 – Input2

- It is used to detect the differences in similar images.



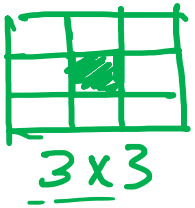
https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html

- Practical applications:
  - Differentiate between human, vehicles etc. from the stationary background.

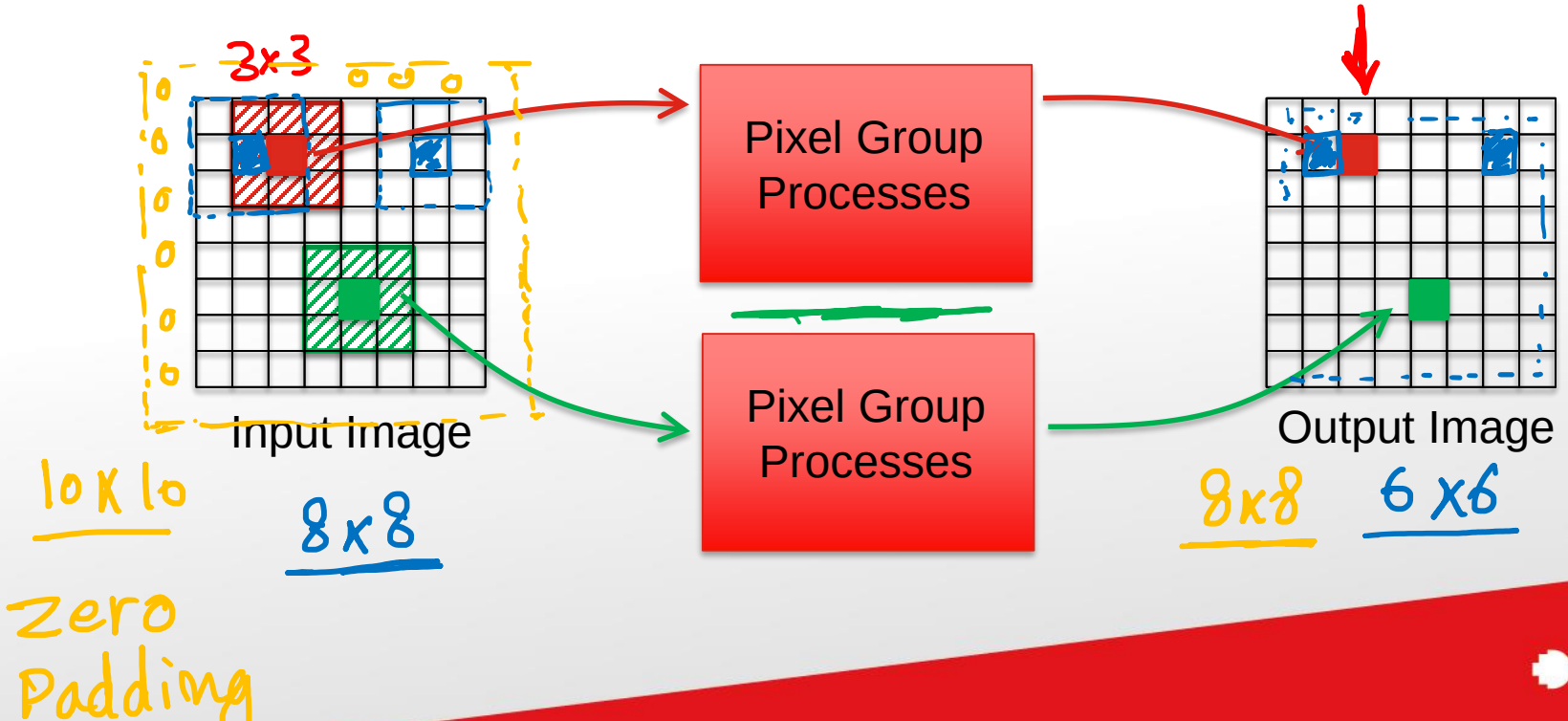
Content

- Introduction to Robotic Vision & Image Processing
- The Digital Image
- Pixel Point Processing
- Pixel Group Processing
- Geometric Transformation
- Feature Extraction

Pixel Group Processing



- Pixel Group Processing operates on a group of pixels surrounding a center pixel.
- The adjoining pixels provide useful information about the area being processed.
- This is also called "Spatial Filtering".



Pixel Group Processing

- The spatial filtering is implemented through a process called spatial convolution.
- This is quite straightforward:

3x3 Pixel group

120 $I(x-1, y-1)$	70 $I(x, y-1)$	70 $I(x+1, y-1)$
$I(x-1, y)$ 118	$I(x, y)$ 75	$I(x+1, y)$
$I(x-1, y+1)$	$I(x, y+1)$	$I(x+1, y+1)$

X (element wise)

Mask

1 W_{11}	0 W_{12}	-1 W_{13}
1 W_{21}	0 W_{22}	-1 W_{23}
W_{31}	W_{32}	W_{33}

→ Then sum up all the elements.

120x1 + 0 - 70 + 118 + 0 + ····

output

Intensity of pixels surrounding the center pixel (x, y)

Weights for each pixel (a.k.a. convolution mask)

- Or:

$$O(x,y) = \begin{aligned} & W_{11}I(x-1,y-1) + W_{12}I(x,y-1) + W_{13}I(x+1,y-1) \\ & W_{21}I(x-1,y) + W_{22}I(x,y) + W_{23}I(x+1,y) \\ & W_{31}I(x-1,y+1) + W_{32}I(x,y+1) + W_{33}I(x+1,y+1) \end{aligned}$$

Low Pass Spatial Filter

- One simple example of the convolution mask would be low pass filter:

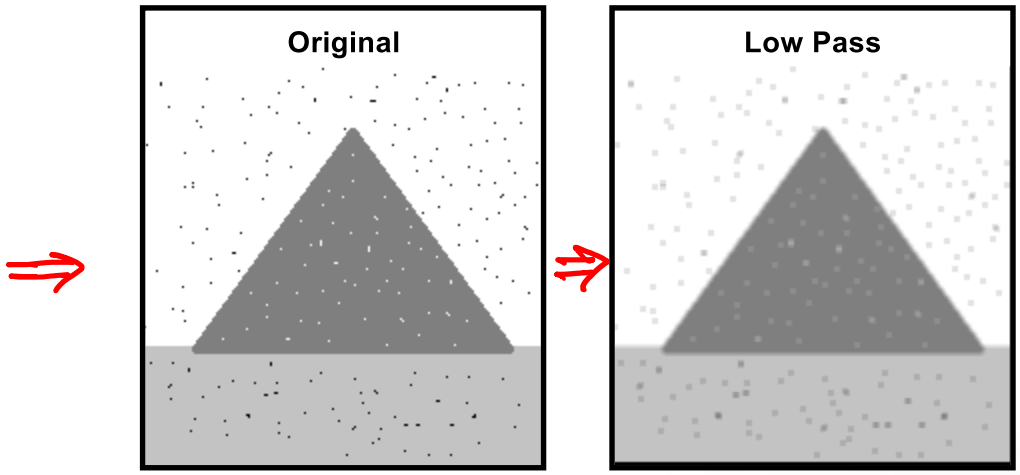
$$\begin{bmatrix} I_{11} & I_{12} & I_{13} \\ I_{21} & I_{22} & I_{23} \\ \cdots & \cdots & \cdots \end{bmatrix} \times \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = 1/9\,(I_{11} + I_{12} + I_{13} + \cdots \cdots \cdots)$$

averaging

- Note that all the weights add up to 1.
- As its name suggest, the low pass filter is useful to remove high-frequency components (e.g. noise) in the image.
  - Low frequency components pass through without much effect.
  - High frequency components get blurred.

Low Pass Spatial Filter

- Example: A mountain picture with lots of noises.



- After low pass filtering, the noises gets blurred.

MATLAB Codes

- Cameraman example:

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

```
I=double(I);
```

```
[m,n]=size(I);
```

Get the image size

```
ILow = zeros(m,n);

for i = 2:m-1
    for j = 2:n-1
        ILow(i,j) = 1/9*I(i-1,j-1)+1/9*I(i-1,j)+1/9*I(i-1,j+1)...
            +1/9*I(i,j-1)+1/9*I(i,j)+1/9*I(i,j+1)...
            +1/9*I(i+1,j-1)+1/9*I(i+1,j)+1/9*I(i+1,j+1);
    end
end

ILow = uint8(ILow);
figure,imshow(ILow)
title('Low Pass')
```

MATLAB Codes

- Cameraman example:

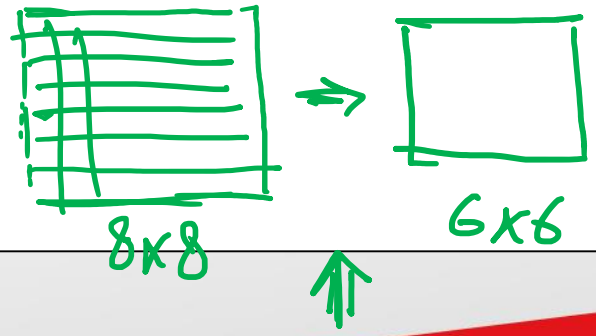
```
I = imread('cameraman.tif');
figure, imshow(I)
title('Original')
```

```
I=double(I);
```

```
[m,n]=size(I);
```

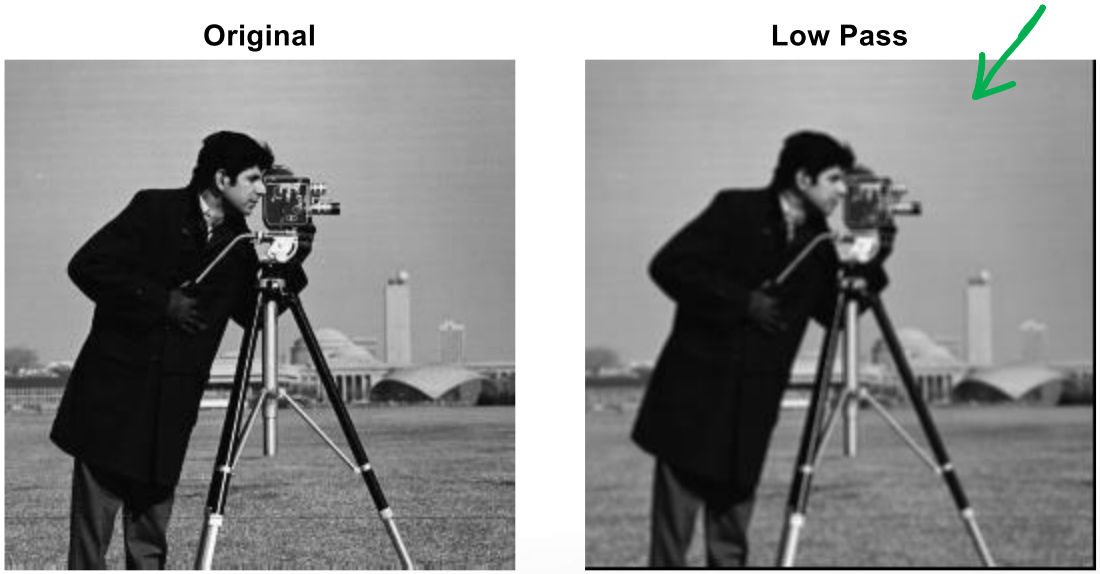
Get the image size

```
ILow = zeros(m,n);

for i = 2:m-1
    for j = 2:n-1
        ILow(i,j) = 1/9*I(i-1,j-1)+1/9*I(i-1,j)+1/9*I(i-1,j+1)...
            +1/9*I(i,j-1)+1/9*I(i,j)+1/9*I(i,j+1)...
            +1/9*I(i+1,j-1)+1/9*I(i+1,j)+1/9*I(i+1,j+1);
    end
end

ILow = uint8(ILow);
figure,imshow(ILow)
title('Low Pass')
```



MATLAB Codes

- The outcome of low pass filtering is as follows:



- Sample values:

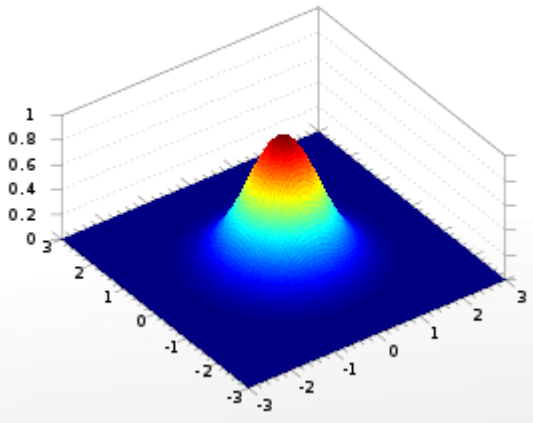
	35	36	37
49	171	171	173
50	173	170	170
51	173	169	172

	35	36	37
49	172	172	172
50	171	171	171
51	171	171	171

Gaussian Filter

- The "simple average" filter shown just now gives equal weight to all pixels.
- Another possibility is to give stronger weight to the centre pixel, then reduce the weights as we go further from the center.
- This can be achieved using a "Gaussian Filter".

$$\begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix}$$

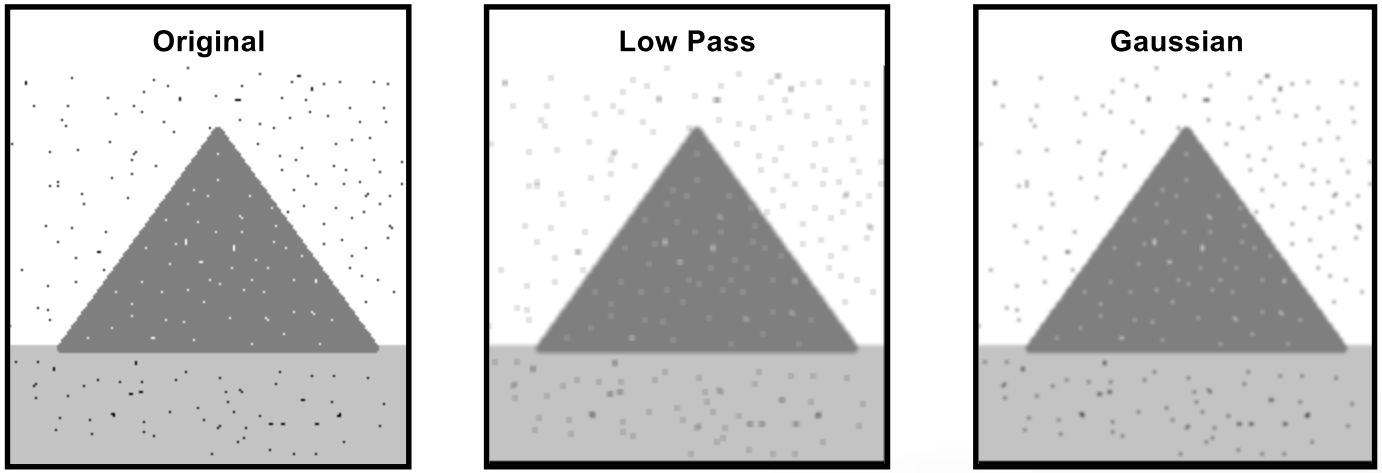


https://az.wikipedia.org/wiki/%C5%9E%C9%99kil:Gaussian_2d.png

- The weights also add up to 1, just as for the simple average filter.

Gaussian Filter

- Example: A mountain picture with lots of noises.



- The Gaussian filter also blurs out the noises.

MATLAB Codes

- Cameraman example:

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

```
I=double(I);
```

```
[m,n]=size(I);
```

Get the image size

```
Igauss = zeros(m,n);

for i = 2:m-1
    for j = 2:n-1
        Igauss(i,j) = (I(i-1,j-1)+2*I(i-1,j)+I(i-1,j+1)...
            +2*I(i,j-1)+4*I(i,j)+2*I(i,j+1)...
            +I(i+1,j-1)+2*I(i+1,j)+I(i+1,j+1))/16;
    end
end

Igauss = uint8(Igauss);
figure,imshow(Igauss)
title('Gaussian')
```

MATLAB Codes

- The outcome of Gaussian filtering is as follows:

Original

Low Pass

Gaussian

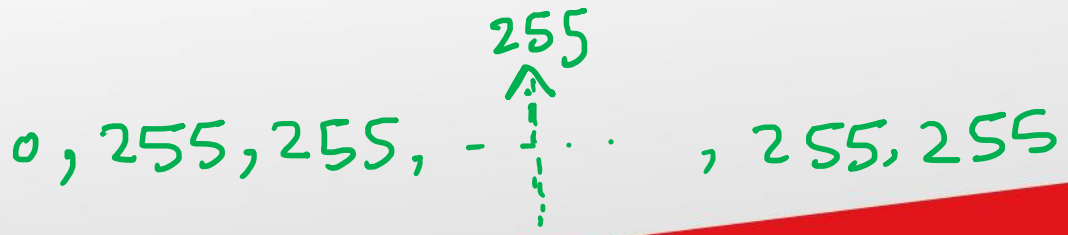
- Sample values:

	35	36	37
49	171	171	173
50	173	170	170
51	173	169	172

	35	36	37
49	171	172	172
50	171	171	171
51	172	171	172

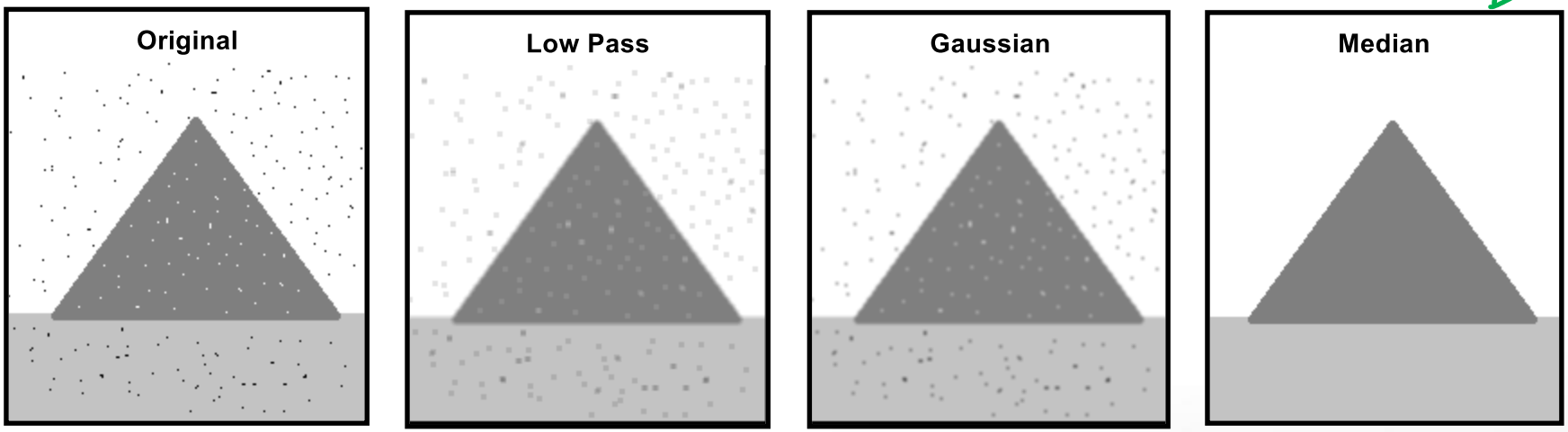
Median Filter

- Pixel Group Processing doesn't necessarily have to be done in the way shown previously, i.e. multiplication with weights then sum up.
- Nonlinear processing is also possible!
- For e.g. it is well known that averaging can be biased due to extreme outliers.
  - E.g. average of 255, 255, 255, 0, 255, 255, 255, 255, 255 = 227.
  - I.e. a group of 3x3 pixels with 8 white pixels and 1 black outlier is averaged out and becomes not so white.
- Is there any better way to remove the outlier?
  - Yes, get the median instead of average!
  - Median of 255, 255, 255, 0, 255, 255, 255, 255, 255 = 255!



Median Filter

- Example: A mountain picture with lots of noises.



- As can be seen, the Median filter almost completely removes the noise!

MATLAB Codes

- Cameraman example:

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

```
I=double(I);
```

```
[m,n]=size(I);
```

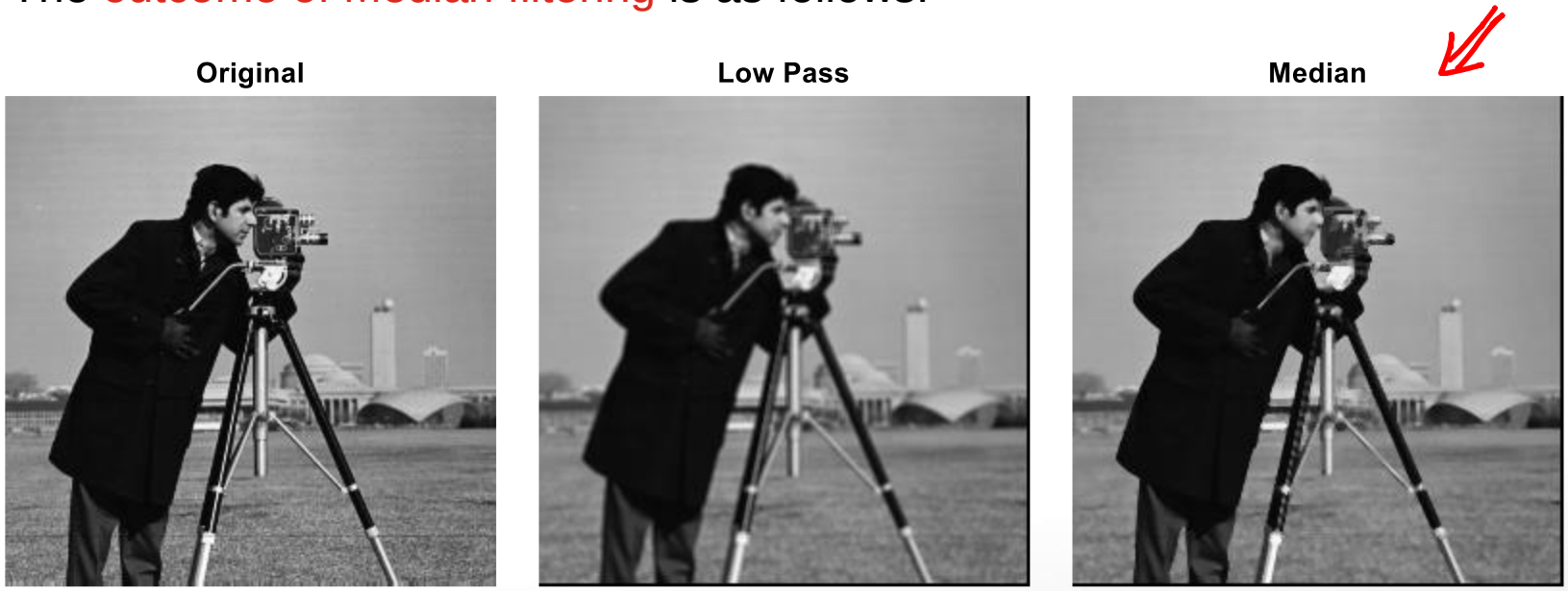
Get the image size

```
Imedian = zeros(m,n);

for i = 2:m-1
    for j = 2:n-1
        Imedian(i,j) = median([I(i-1,j-1),I(i-1,j),I(i-1,j+1)...
            ,I(i,j-1),I(i,j),I(i,j+1)...
            ,I(i+1,j-1),I(i+1,j),I(i+1,j+1)]);
    end
end

Imedian = uint8(Imedian);
figure,imshow(Imedian)
title('Median')
```

MATLAB function for getting median

MATLAB Codes

- The outcome of Median filtering is as follows:



- Sample values:

	35	36	37
49	171	171	173
50	173	170	170
51	173	169	172

	35	36	37
49	171	172	172
50	171	171	171
51	171	171	171

High Pass Spatial Filter

- Opposite to Low Pass Filter, the High Pass Spatial Filter is to enhance the high frequency components.
- The convolution mask is:

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

- Note that the weights also sum up to 1.
- Interpretation:
  - High influence by the center pixel, while the neighbours act to oppose it.
  - If the center pixel is very different from the neighbours, which indicates sharp transition in grey level, the surrounding pixel effect becomes negligible.
  - If the center pixel is similar to the neighbours, then the output becomes more like an average of all pixels.

High Pass Spatial Filter

- E.g. sharp transition in grey level:

dark bright

3x3

$$\begin{bmatrix} 10 & 250 & 250 \\ 10 & 250 & 250 \\ 10 & 250 & 250 \end{bmatrix} \text{ X (Element -wise) } \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \rightarrow$$

970, which saturates at 255

- E.g. similar grey level:

$$\begin{bmatrix} 150 & 155 & 155 \\ 150 & 155 & 155 \\ 150 & 155 & 155 \end{bmatrix} \text{ X (Element -wise) } \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \rightarrow$$

161

- High pass filter makes the image sharper!

MATLAB Codes

- Cameraman example:

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

```
I=double(I);
```

```
[m,n]=size(I);
```

Get the image size

```
IHigh = zeros(m,n);

for i = 2:m-1
    for j = 2:n-1
        IHigh(i,j) = -1*I(i-1,j-1)-1*I(i-1,j)-1*I(i-1,j+1)...
            -1*I(i,j-1)+9*I(i,j)-1*I(i,j+1)...
            -1*I(i+1,j-1)-1*I(i+1,j)-1*I(i+1,j+1);
    end
end

IHigh = uint8(IHigh);
figure,imshow(IHigh)
title('High Pass')
```

MATLAB Codes

- The outcome of High Pass filtering is as follows:

Original

Low Pass

High Pass

- Sample values:

	35	36	37
49	171	171	173
50	173	170	170
51	173	169	172

	35	36	37
49	166	163	184
50	192	158	161
51	188	147	177

Sharpening

- Apart from using high pass filter, another way of sharpening image is by using the following convolution mask:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \begin{bmatrix} -1/9 & -1/9 & -1/9 \\ -1/9 & 17/9 & -1/9 \\ -1/9 & -1/9 & -1/9 \end{bmatrix}$$

*2 low pass

- Interpretation:
  - (Original * 2) then minus away the "blurred" average → Sharp!

MATLAB Codes

- Cameraman example:

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

```
I=double(I);
```

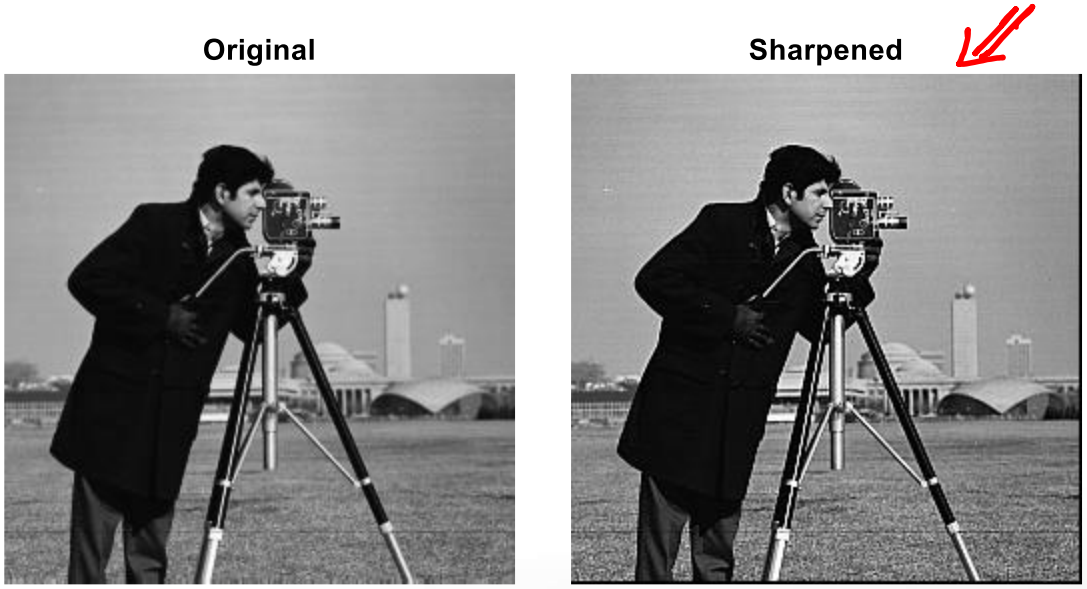
```
[m,n]=size(I);
```

Get the image size

```
Isharp = zeros(m,n);

for i = 2:m-1
    for j = 2:n-1
        Isharp(i,j) = -1/9*I(i-1,j-1)-1/9*I(i-1,j)-1/9*I(i-1,j+1)...
            -1/9*I(i,j-1)+17/9*I(i,j)-1/9*I(i,j+1)...
            -1/9*I(i+1,j-1)-1/9*I(i+1,j)-1/9*I(i+1,j+1);
    end
end

Isharp = uint8(Isharp);
figure,imshow(Isharp)
title('Sharpened')
```

MATLAB Codes

- The outcome of sharpening is as follows:



- Sample values:

	35	36	37
49	171	171	173
50	173	170	170
51	173	169	172

	35	36	37
49	170	170	174
50	175	169	169
51	175	167	173

Thank you!

Have a good evening.

RMIT UNIVERSITY