Robotics Club IITKanpur

Image Processing

Faizan Siddiqui
Naishadh Parmar

How to start with computer vision ?

1. We have ‘OpenCV’
2. Open source image processing library for c++ / python

Representation of Image in a program

1. Prefer Python to use opencv (easy to use)
2. Represented as a numpy array in Python
3. Represented as `cv::Mat` in C++

Get Some hands on !

- Open an image using : `cv2.imread()`
- Show the image using : `cv2.imshow()`
- See the shape of the image using `X.shape` (in Python)
- See the original dimension of the image

Colorspace can be changed

There are a number of colorspaces such as:

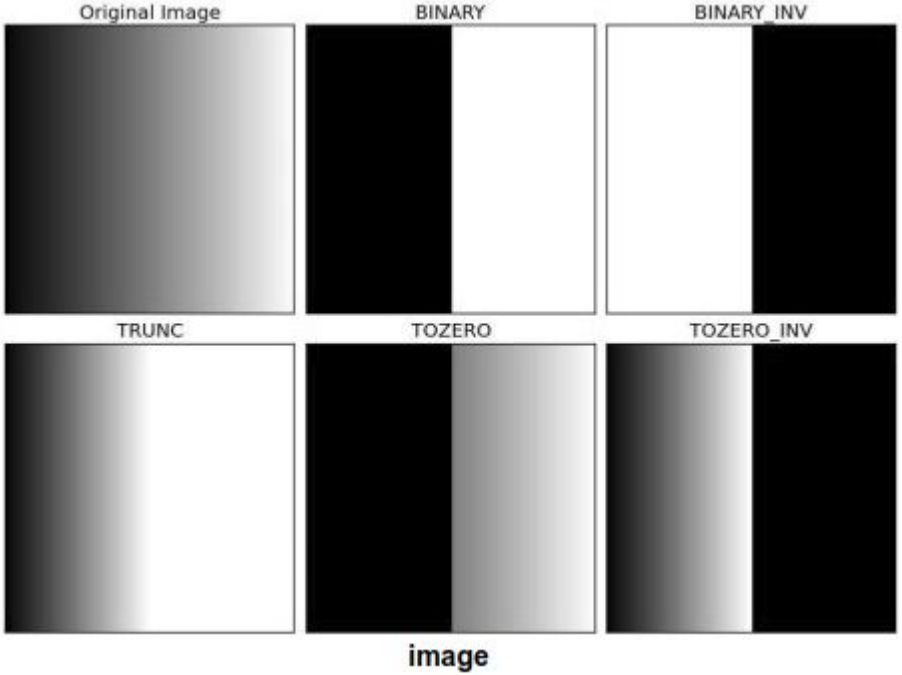
- Blue, Green, Red (BGR)
- Hue, Saturation, Value (HSV)
- Grayscale
- Others which are less used

Shapes can be drawn

You can draw geometric figures on images either for demonstration purposes or otherwise

1. Rectangles
2. Circles
3. Ellipse
4. Polylines, Polygons

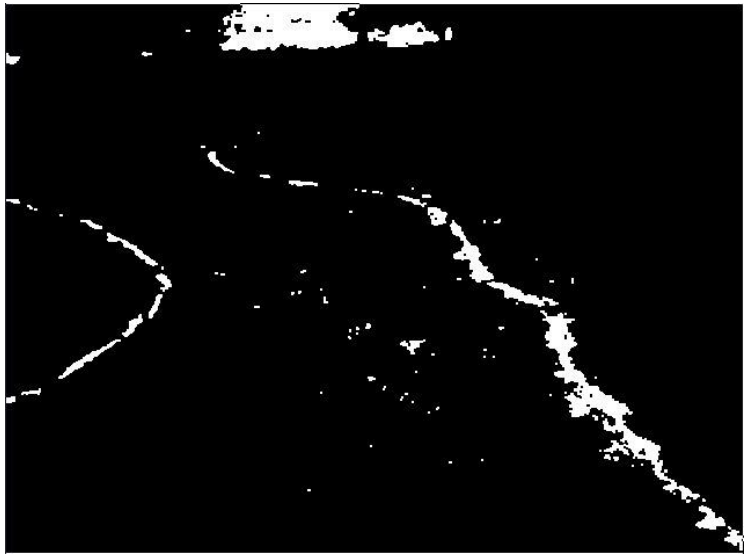
Threshold

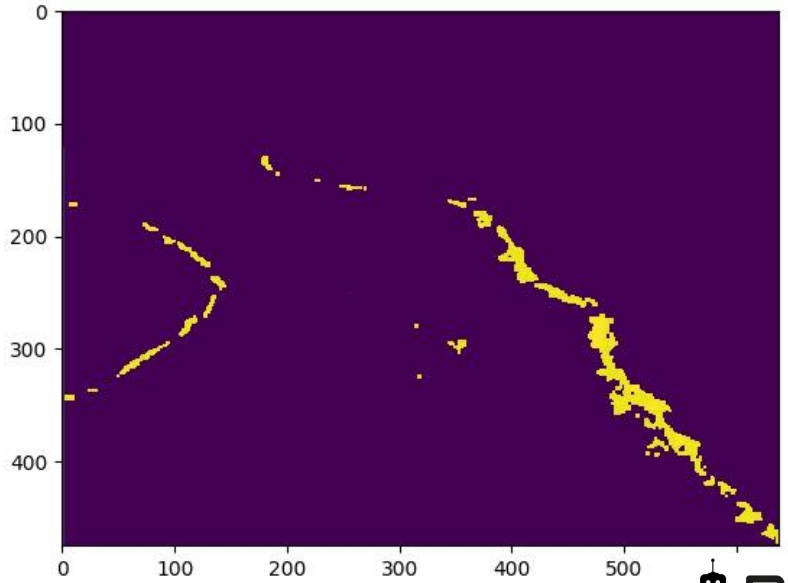


Robotics Club IITKanpur

Denoising and image ?

Seems tough ?





(Some) Methods that can be used to denoise :

1. Erode / Dilate
2. Blurring (Gaussian, median, bilateral)

Erode / Dilate

Erode (left) and Dilate (right)




What can be done using filters ?

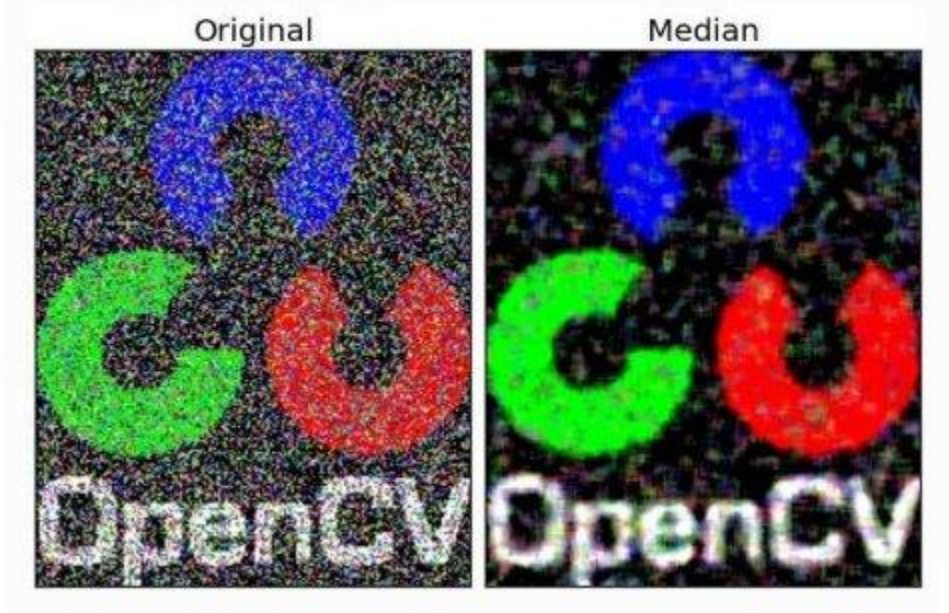
1. Blurring
2. Edge detections
3. Face detections? (maybe)

Gaussian blur :

Kernel size was 9 (it should be an odd number)

Here it’s just that the kernel should be symmetrically placed on each pixel

Median Blur :



Robotics Club IITKanpur

Line Detection filters

1. Which kind of filter is capable to do so?
2. Suggestions?

This is a possible line detection filter

More are possible (dimensionally as well)

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	−1	2
−1	2	−1
2	−1	−1

+45°

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

2	−1	−1
−1	2	−1
−1	−1	2

−45°

Robotics Club IITKanpur

Edge detections :

1. What can be used to detect edges ?
2. Would the filter in the last slide work ?
3. Why / Why not ?

More sophisticated filters for edge detection :

What would X filter do ?

What Would Y filter do ?

-1	0	+1
-2	0	+2
-1	0	+1

x filter

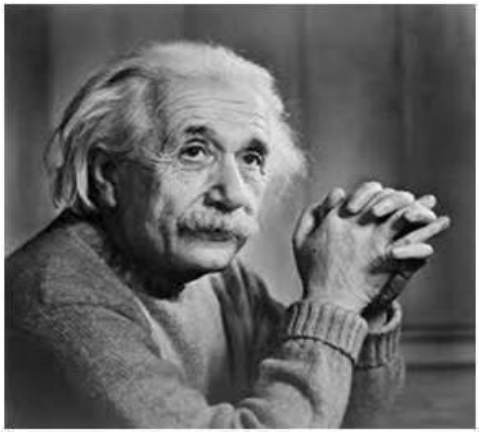
+1	+2	+1
0	0	0
-1	-2	-1

y filter

Robotics Club IITKanpur

This is what their application looks like :

Original : X operator : Y Operator



Robotics Club IITKanpur

Gradient of an image :

1. See image as a discrete valued function
2. Find d/dx
3. Find d/dy
4. Laplacian?

$$\nabla \cdot \nabla f = \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

Laplacian (filter and output)

Our old friend :P

	1	
1	-4	1
	1	

		1		
	2	-8	2	
1	-8	20	-8	1
	2	-8	2	
		1		

Robotics Club IITKanpur

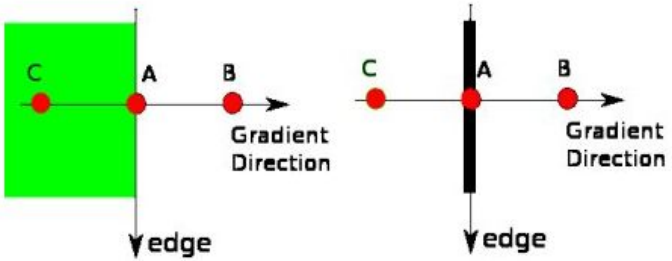
Use built-in functions

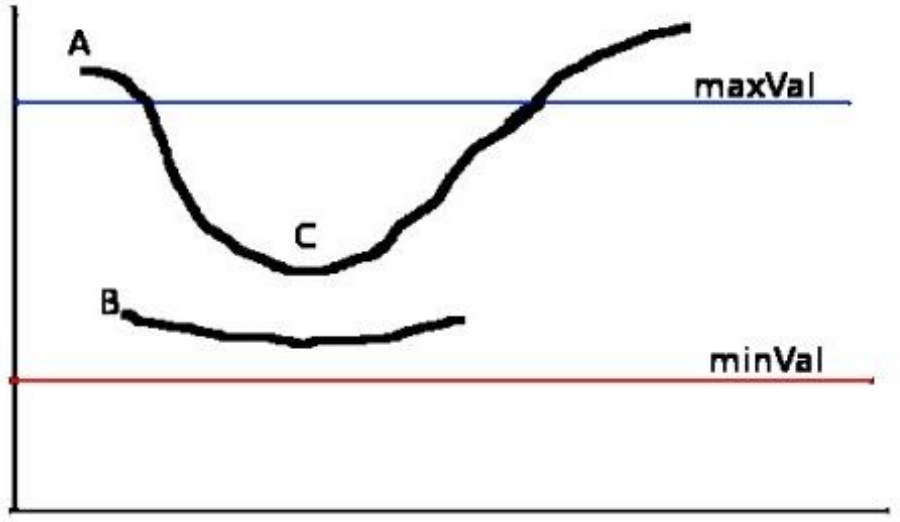
1. `laplacian = cv2.Laplacian(img, cv2.CV_64F)`
2. `sobel_x = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)`
3. `sobel_y = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=5)`
4. No need to write these Kernels on our own
5. However feel free to improvise (and send a PR to OPENCV :P)

A better edge detector - Canny

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$





Robotics Club IITKanpur

Finally, face detection

This uses the Haar Feature-based Cascade Classifiers

THANK YOU

Robotics Club IITKanpur