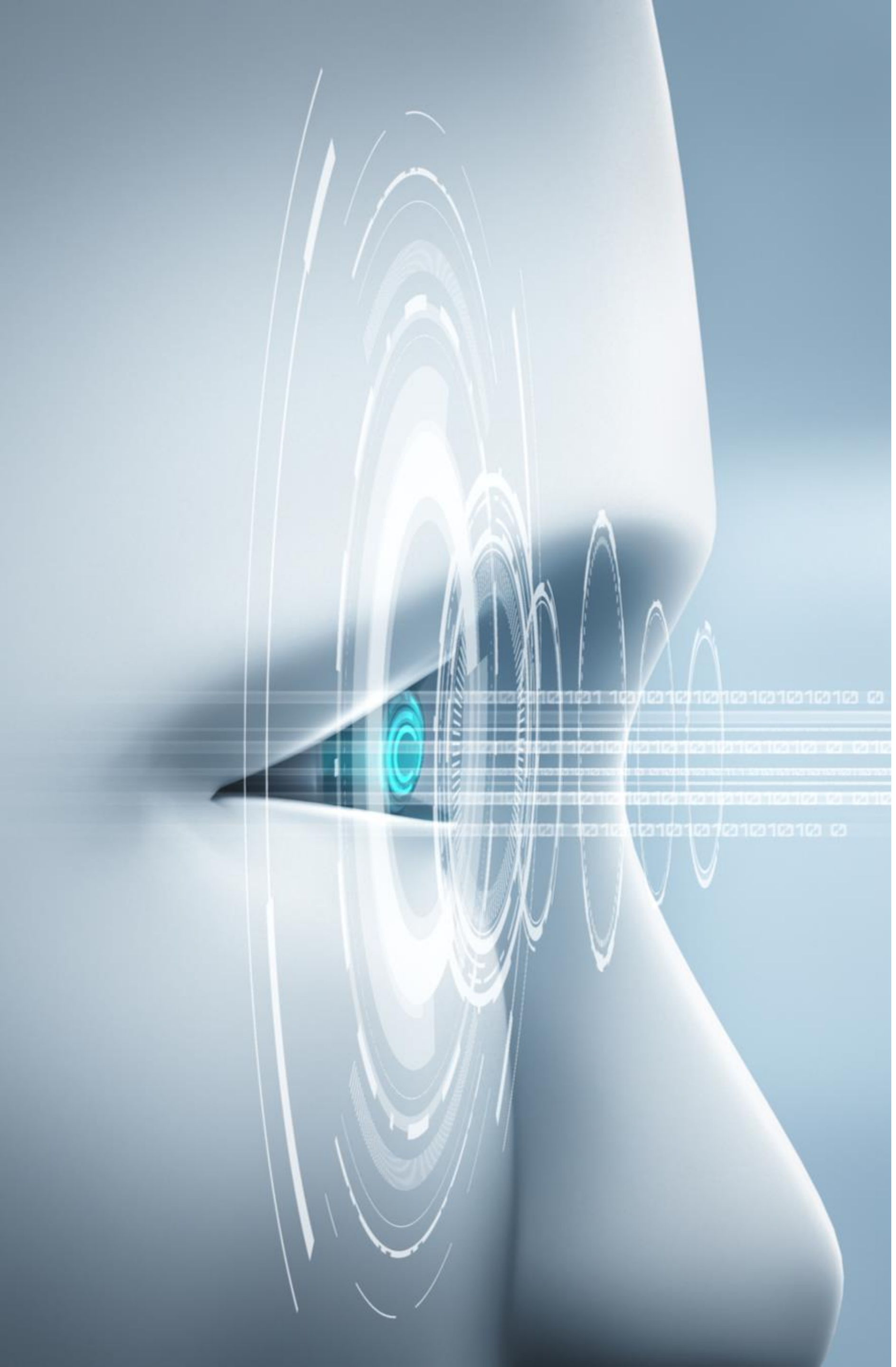


Computer Vision Applications

BY QUADEER SHAIKH

About me

Work Experience

- Risk Analyst
  - Morgan Stanley (Jan 2023 – Present)
- Data Science Intern
  - AkzoNobel Coatings International B.V. Netherlands (Feb 2022 – Dec 2022)
- Data Science Intern
  - EzeRx Health Tech Pvt. Ltd. (Jan 2022 – July 2022)
- Associate Engineer
  - Tata Communications Ltd. (July 2019 – Aug 2020)
- Network Automation and Analysis Engineer Intern
  - Cisco (June 2018 – July 2018)

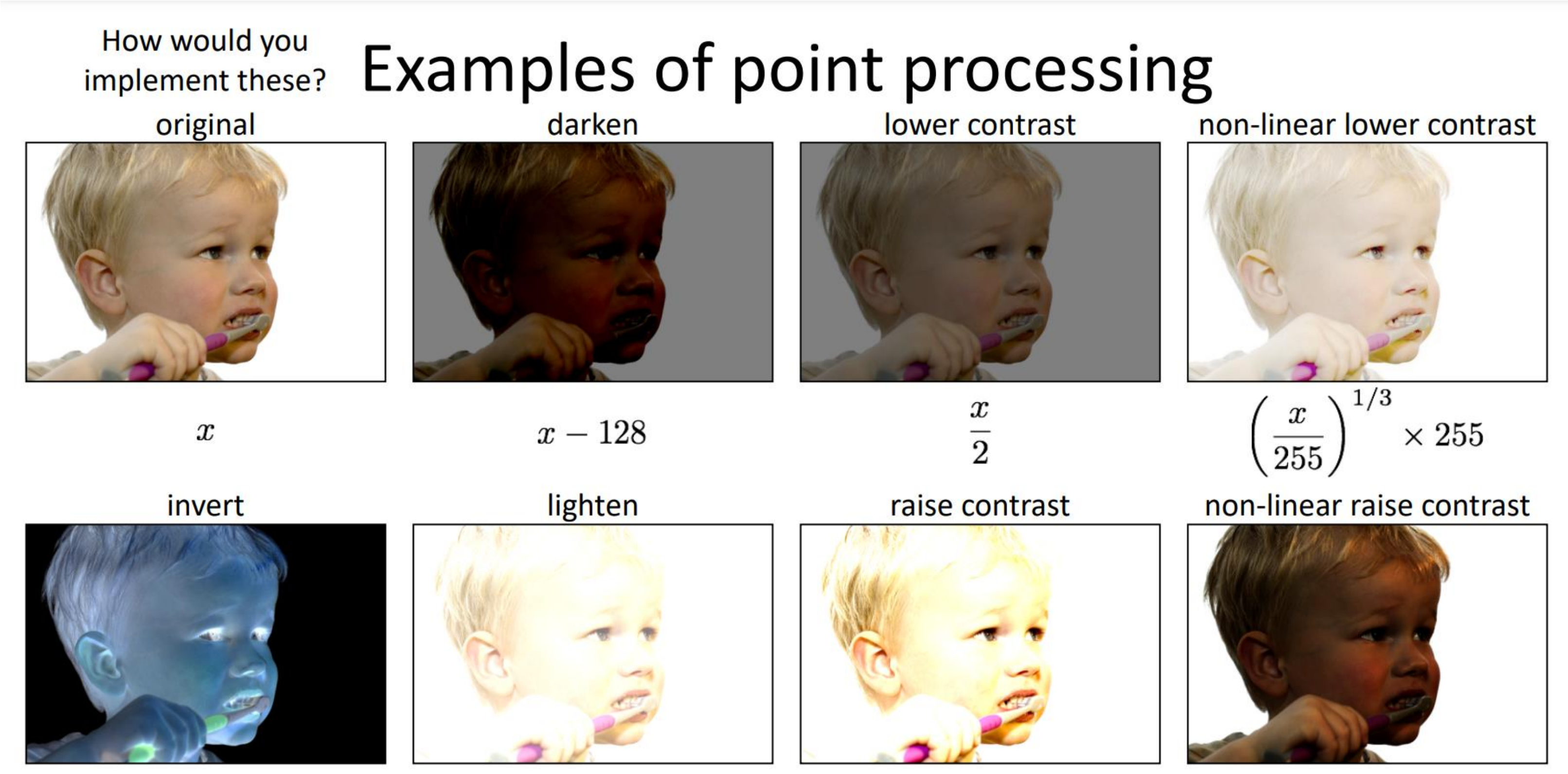
Education

- M.Tech – Artificial Intelligence
  - NMIMS (2021 - 2023, currently pursuing)
- B.E. – Computer Engineering
  - Mumbai University (2015 - 2019)

Image Operations

1. Pixel based operations
    1. Contrast Stretching
    2. Thresholding
    3. Inverting/Negative Images
    4. Erosion and Dilation
    5. Bitwise operations, etc.
2. Regions based operations
    1. Contour Detection
    2. **Edge Detection**
    3. **Line Detection, etc.**

Pixel Based Operations



A brief discussion on Self Driving Cars

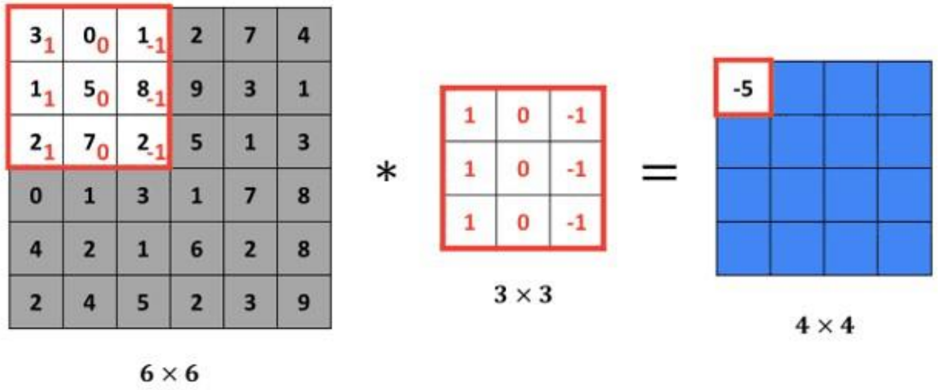
- A self driving car/autonomous car/robotic car is a car that is capable of travelling without human input
- Use sensors to perceive their surrounding: optical and thermographic cameras, radar, lidar, ultrasound, GPS, odometry, etc.
- Focuses on
  - Identification of proper navigation paths
  - Strategies for managing traffic controls and obstacles
  - Human factors

Lane Detection using Edge Detection and Line Detection

Edge Detection

- Edge detection includes a variety of mathematical methods that aim at identifying edges, curves in a digital image at which the image brightness changes sharply or, more formally, has discontinuities.
- Edge detection is a fundamental tool in image processing and computer vision, particularly in the areas of feature detection and feature extraction
- Achieved by applying convolution operations using filters

Convolution Operation using Filter

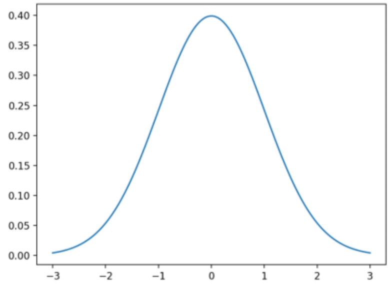


$$3\times 1+1\times 1+2\times 1+0\times 0+5\times 0+7\times 0+1\times -1+8\times -1+2\times -1=-5$$

$$\underbrace{h[m,n]}_{\text{output}}=\sum_{k,l}\underbrace{g[k,l]}_{\text{filter}}\underbrace{f[m+k,n+l]}_{\text{image (signal)}}$$

Filter Example: Gaussian Filter

- Named after Carl Fredrich Gauss
- Kernel values sampled from 2D Gaussian Function
- The weight reduces as drift away from the center pixel
- Used for reducing noise in an image or for smoothening image
- Other choices of smoothening: box filter, median filter



$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

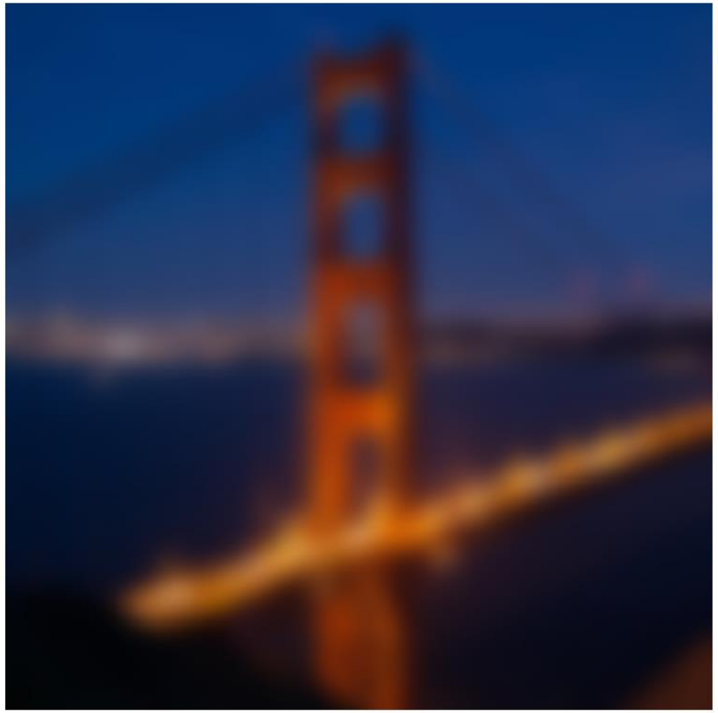
$\frac{1}{16}$ ×

1	2	1
2	4	2
1	2	1

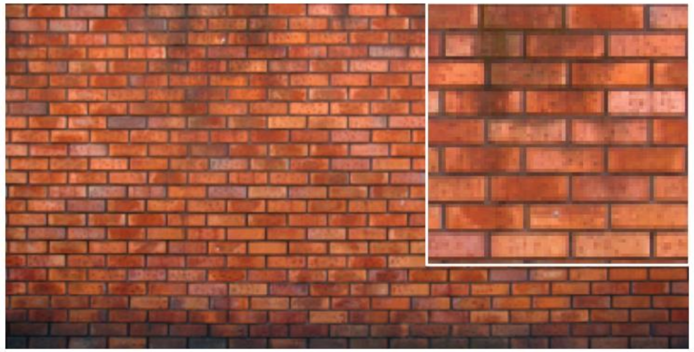
$\frac{1}{273}$ ×

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Example: Gaussian Filter Result

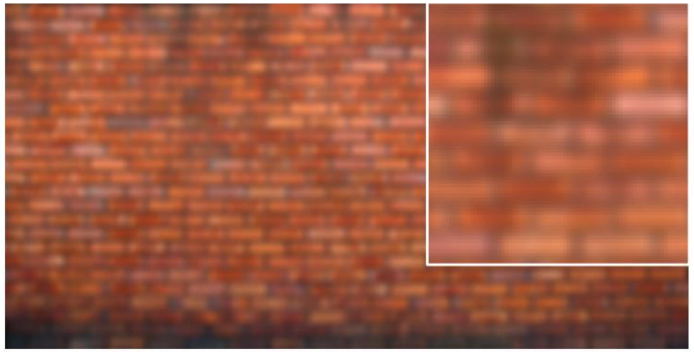


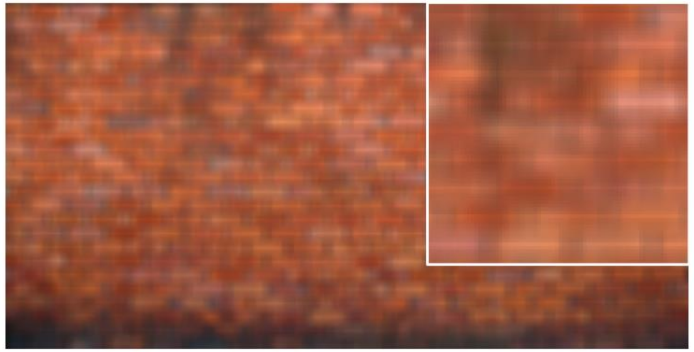
Comparison between smoothening filters



original

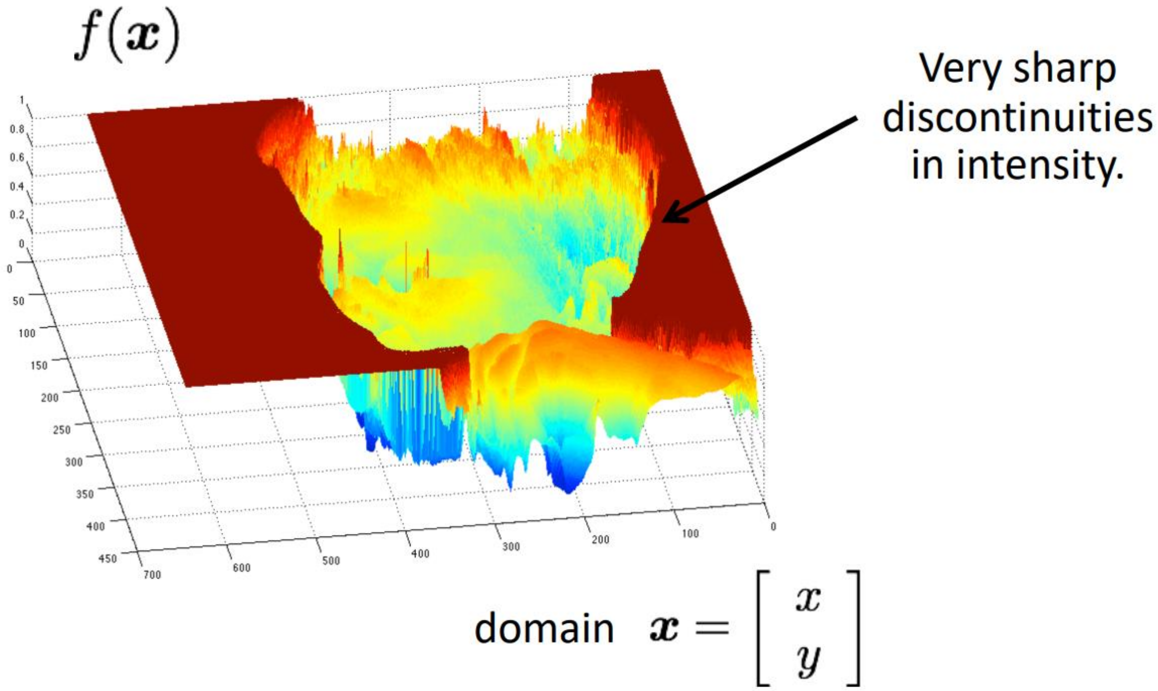
Which blur do you like better?





What are edges in Images ?

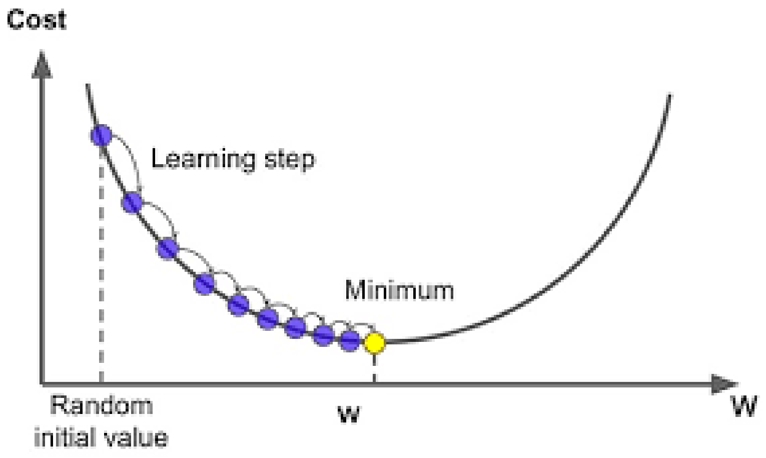
grayscale image



How to determine edges ?

- How do you detect edges/How do you calculate discontinuities in a function ?
- Ans: You take derivatives, as derivatives are large at discontinuities. Since it’s a 2D function we call it gradient and not derivative
- The same analogy is used in Machine Learning for gradient descent
- Since images are 2D discrete signals, we take finite differences

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$



Gradient Filters

Sobel

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

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

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

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

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

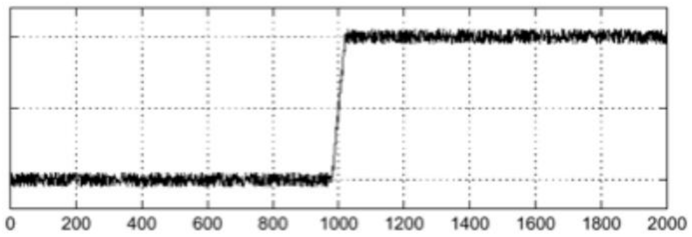
Roberts

0	1
-1	0

1	0
0	-1

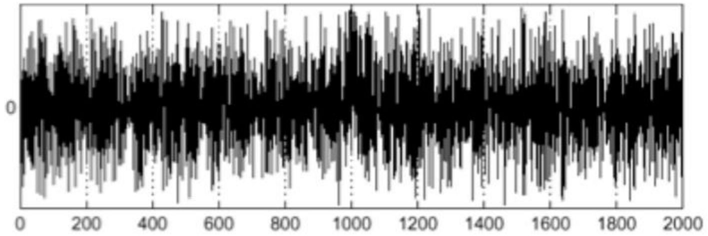
Differentiation of a 1D Signal

intensity plot



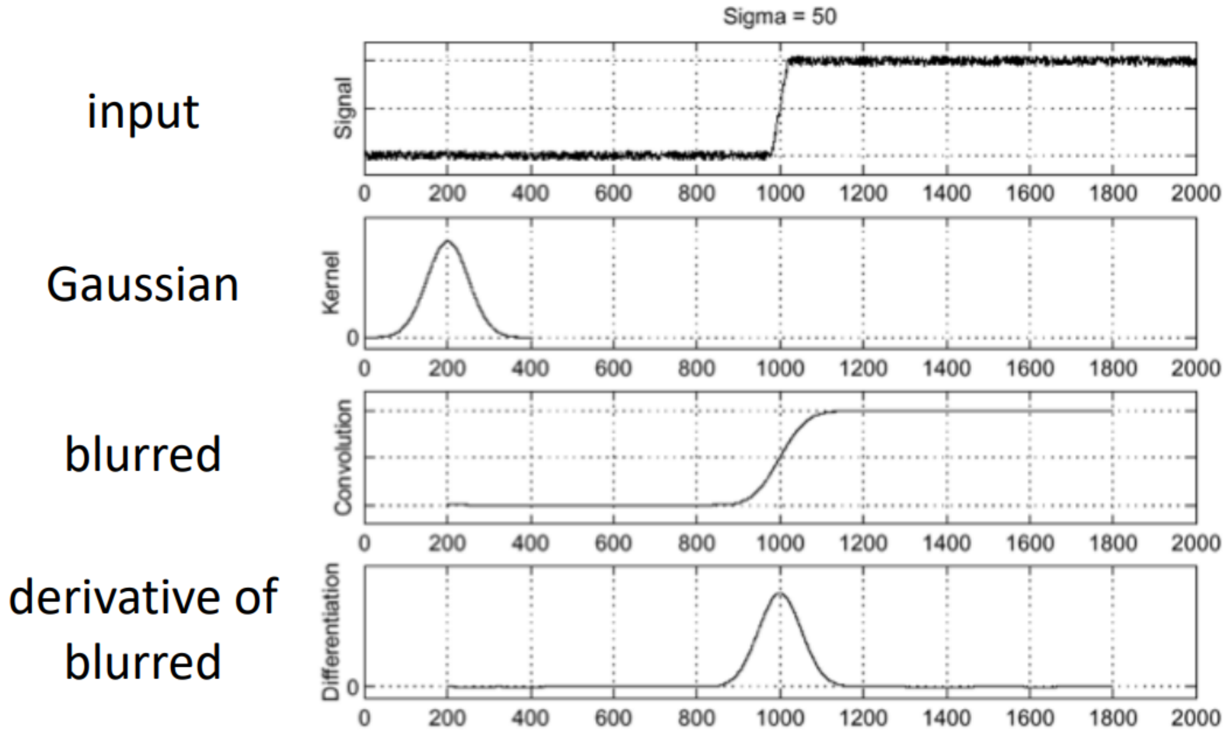
Using a derivative filter:

derivative plot



What’s the problem here?

Derivatives/Gradient are sensitive to noise



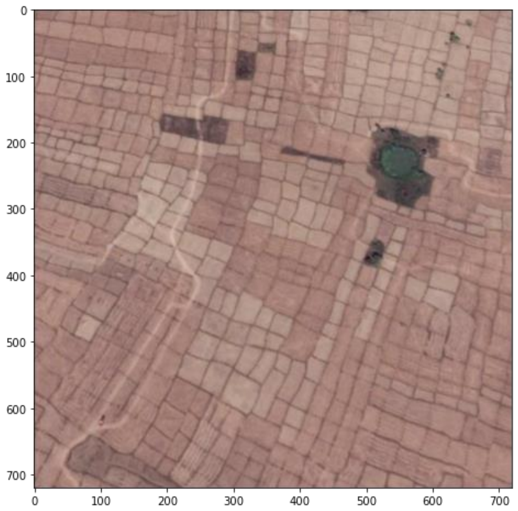
Sobel Filter Example

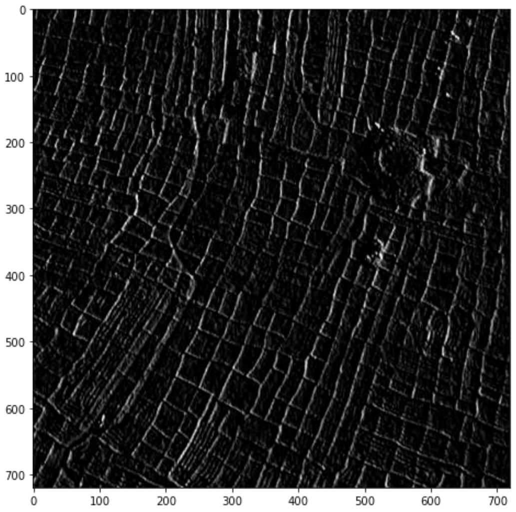
-1	0	1
-2	0	2
-1	0	1

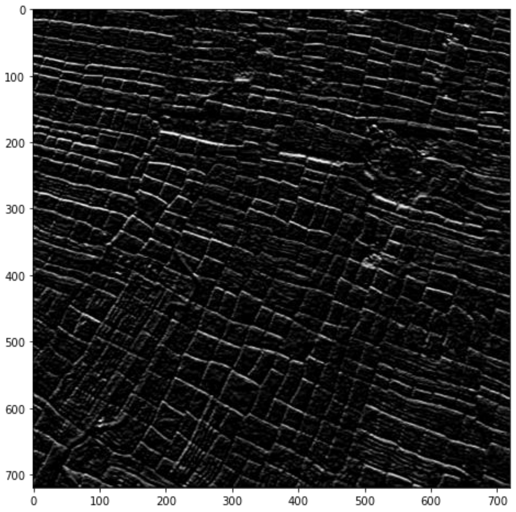
Vertical

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

Horizontal







Canny Edge Detection

1. Convert Image to Grayscale
2. Gaussian Blur
   - Removes noise from the image
3. Determine the Gradients
   - Both horizontal and vertical gradient filters applied and magnitude is calculated to detect pixel intensity changes
4. Non Maximum Suppression
   - Image magnitude in the previous step results in thick edges, the final image should have thin edges
   - Perform a suppression based on comparison of neighbouring pixel values

https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

Canny Edge Detection

5. Double Thresholding

- Result from NMS is not perfect, some edges may not be edges and there is some noise present in the image
- Two thresholds are maintained: low thresh and high thresh e.g. 50 and 150
    1. Areas with pixel values which are less than low thresh are declared as non edges and are set to 0
    2. Areas with pixel values which are greater than high thresh are declared as strong edges and are set to 255
    3. Areas with pixel value between low and high thresh are declared as weak edges and are dealt with in step 6

6. Edge Tracking by Hysteresis

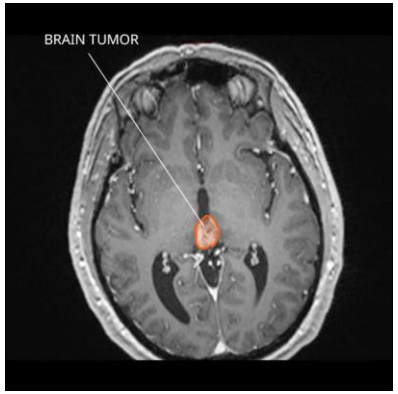
1. Weak edges which are connected to strong edges will be actual edges
2. Weak edges which are not connected to strong edges are removed

https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

Region of Interest

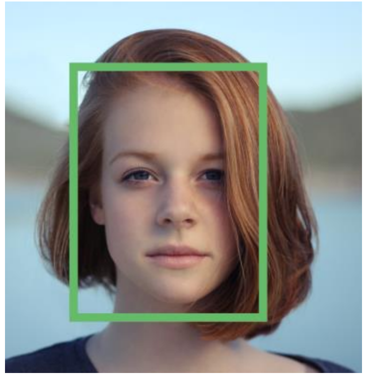
A region of interest (ROI) is **a portion of an image that you want to filter or operate on in some way**. You can represent an ROI as a binary mask image. In the mask image, pixels that belong to the ROI are set to 1 and pixels outside the ROI are set to 0

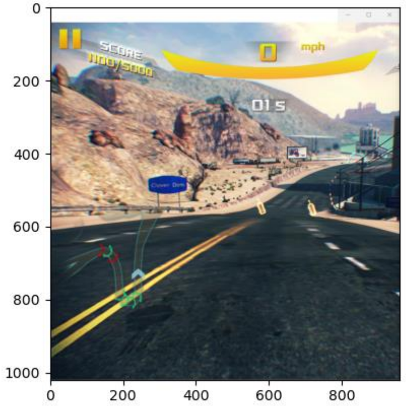
- Region of Interest pixels are denoted using 255 (all 8 binary digits set to 1)
- Pixels out of ROI are denoted using 0 (all 8 binary digits set to 0)
- E.g. Logo of a team on a jersey, tumor in a CT-SCAN of brain, detection of face before detecting masks, etc.

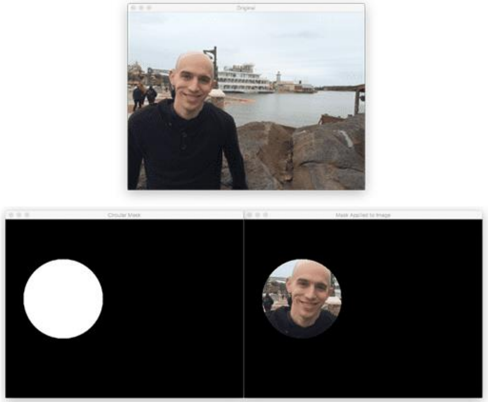


How to find your ROI ?

- Determine ROI using detection algorithms
- Determine ROI using contours
- Determine ROI using graph coordinates
- Determine ROI using Image masking







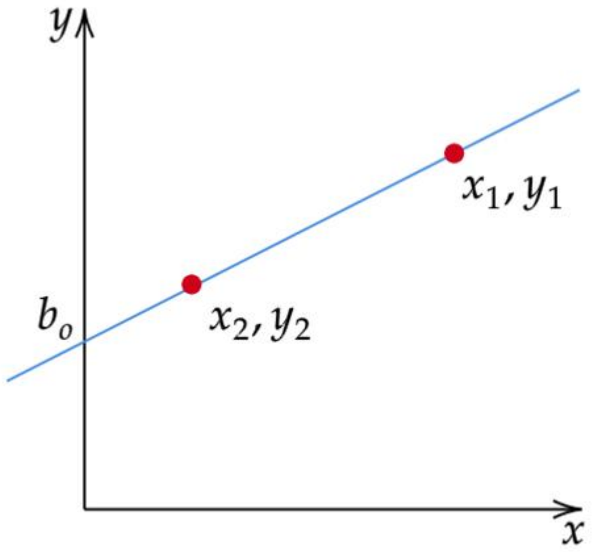
Line Detection

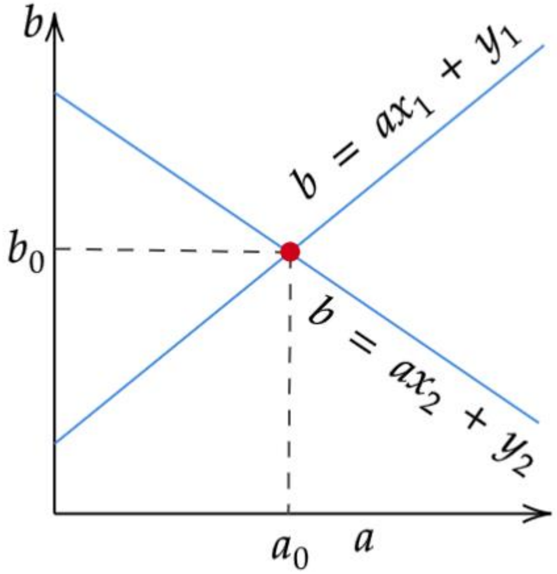
Line detection is an algorithm that takes a collection of n edge points and finds all the lines on which these edge points lie

The most popular line detectors are the Hough transform and convolution-based techniques.

Hough Transform

Equation of a line is y = mx + c
A point in xy coordinate plane is equivalent to a line in hough space or mx coordinate space

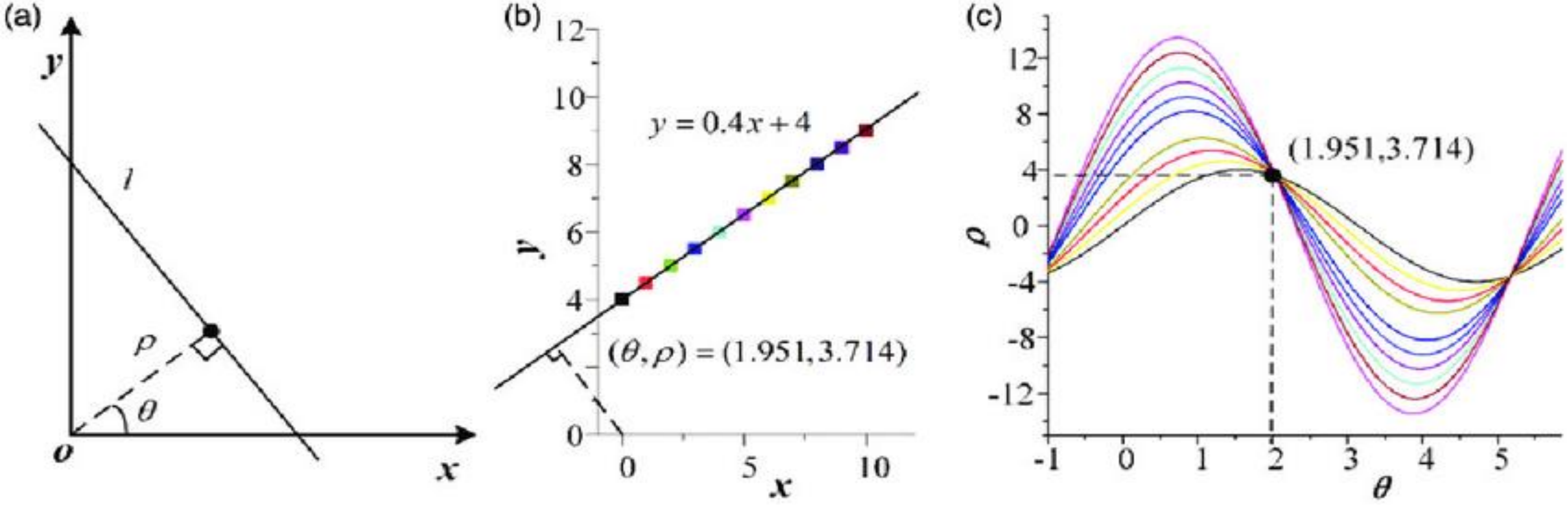




Hough Transform Contd.

Vertical Lines have slope value as infinity, therefore we instead represent the line using polar coordinates

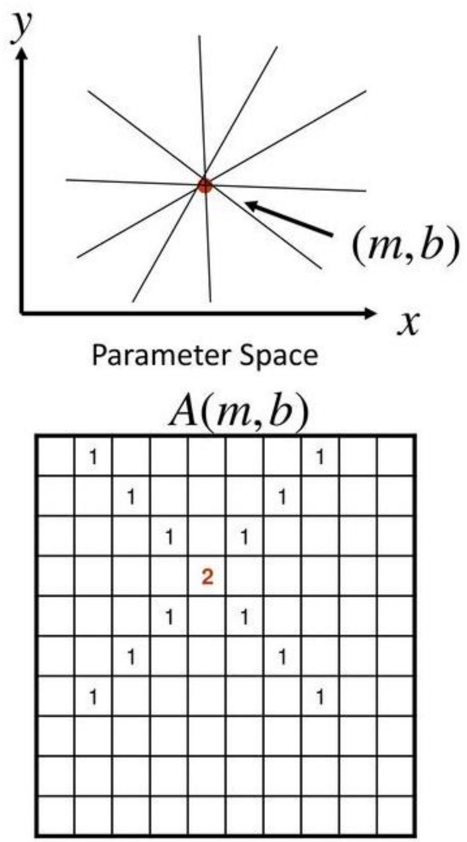
A point in xy coordinate plane is equivalent to a line in hough space or polar coordinate space



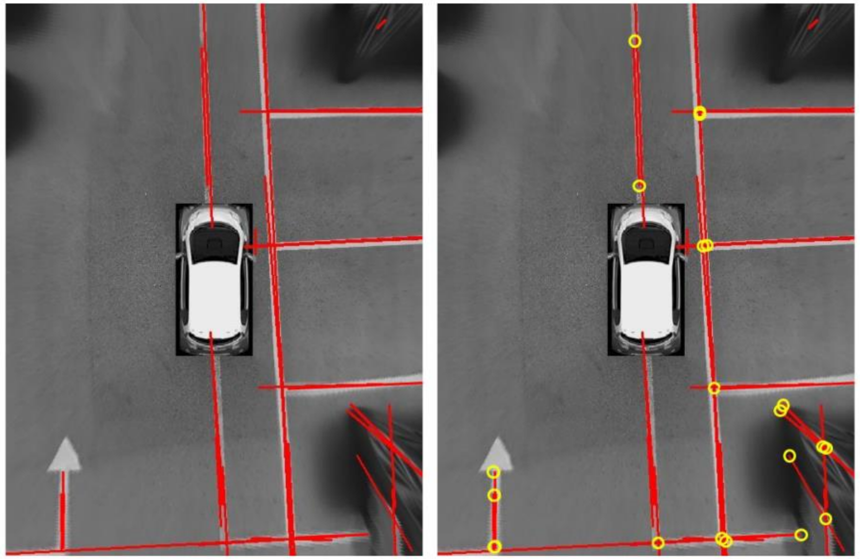
Hough Transform Line Detection Algorithm

1. Extract edges using Canny edge detection algorithm
2. Initialize parameter space p and theta using a 2D array (referred to as accumulator array)
3. Initialize all the values in the accumulator array to zero
4. For each edge pixel
    1. For each value of theta
        1. Calc p = xcos(theta) + ysin(theta)
        2. Increment the value in Accumulator array A at A(p,theta)
5. Find the max value in Accumulator array A, these will be the parameters p and theta for your line.

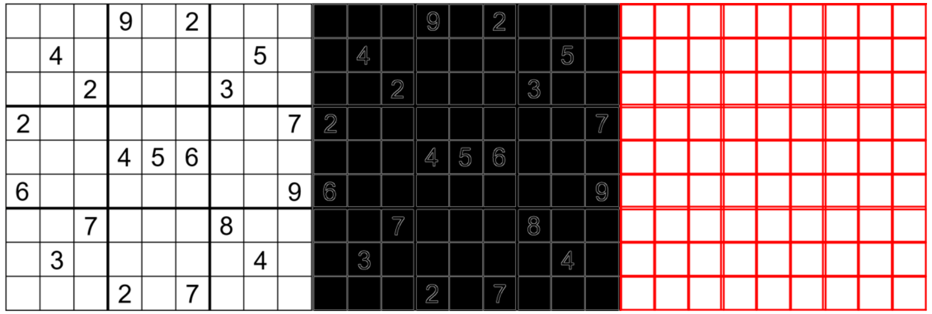
Note: Sometimes they are converted back into m and b parameter space in opencv implementations



Some other Applications of Line Detection



(a) (b)



Classroom Doubts

1. Why do we use line detection followed by edge detection ?
   - Since the edges detected using canny edge detection are of different types (horizontal, vertical, diagonal, wavy, etc.) to detect straight lines we apply line detection algorithm on top of the ROI (edges detected)

2. While averaging params do we need to keep axis=0 ?
   - Yes, I made an error during the classwork exercise you will have to mention axis=0 so that you get an average of both slope and intercept [w b], and not just a single average value (if you do not specify axis=0)

3. Why do we have to apply optimization of lines ? (reduce multiple lines into one, isn't that what we do in Non max suppression?)
   - We perform non max suppression in canny edge detection to suppress thick edges into thinner consistent edges. In line detection multiple lines may get detected depending on the presence of edge pixels present, therefore we perform parameters averaging to reduce multiple parallel lines into one line which is represented using the average parameters.

Thank you

For any queries drop an email at: quadeershaikh15.8@gmail.com