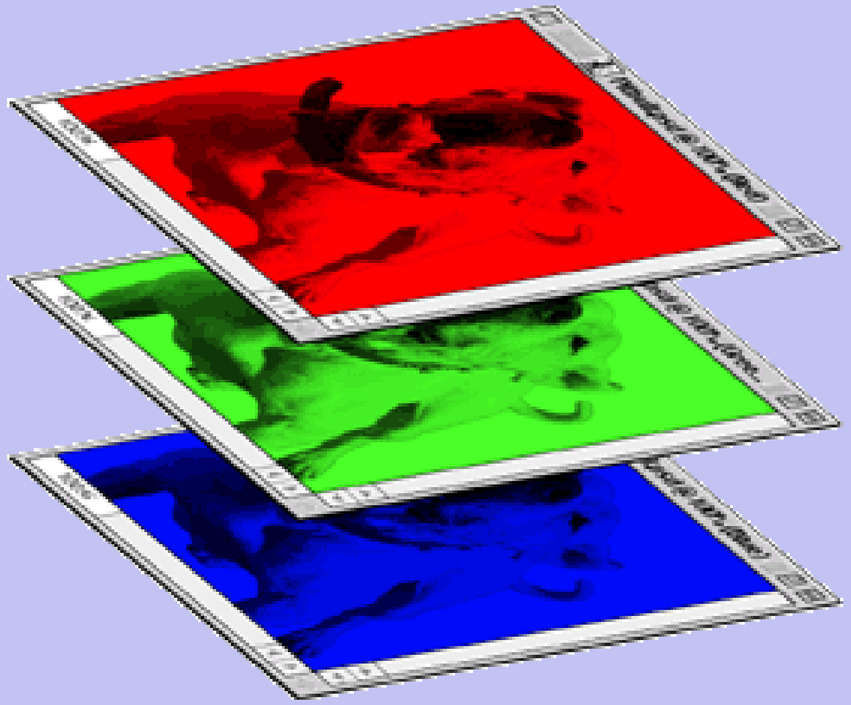
ELECTRONICS CLUB

IMAGE PROCESSING AND COMPUTER VISION

- HOW CAN IMAGE PROCESSING CAN SOMETIMES DO MAGIC ?!

WHAT ARE IMAGES ?

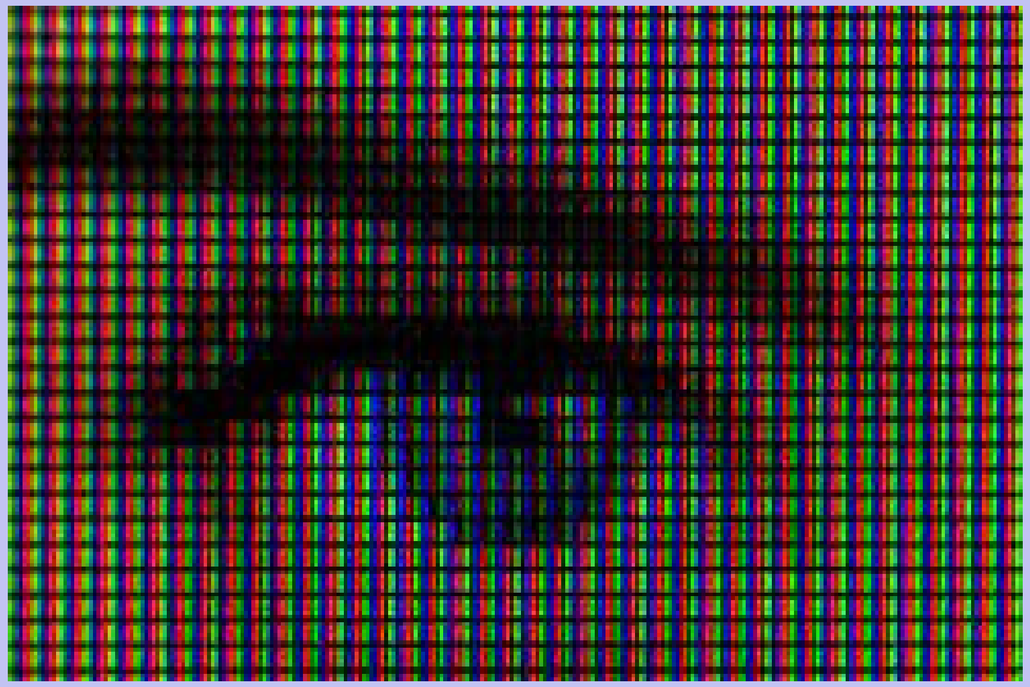
HOW YOUR COMPUTER PERCIEVES IMAGES ?



Red Channel

Green Channel

Blue Channel



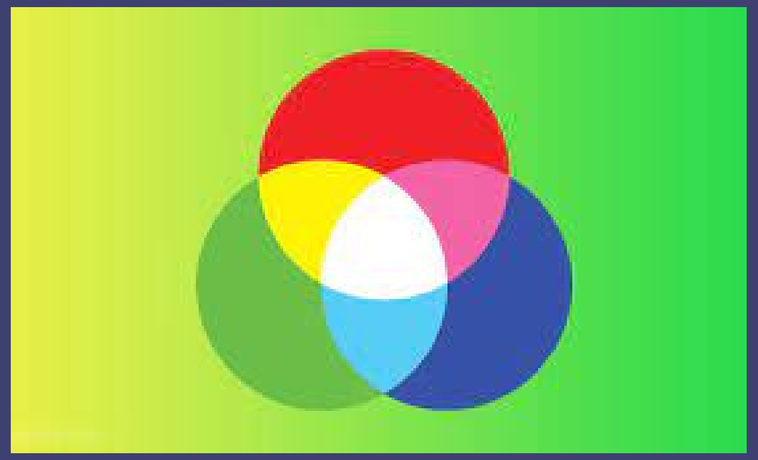
HOW COMPUTER SAVE YOUR MEMORIES :)

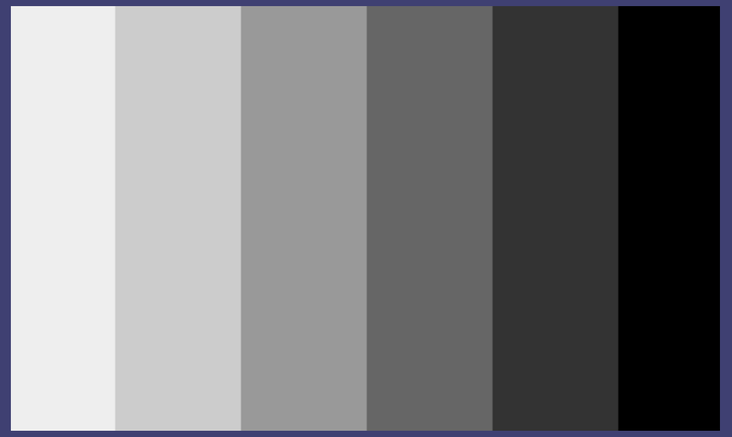
HOW TO ACCESS EACH BGR VALUES OF EACH PIXEL IN AN IMAGE ?

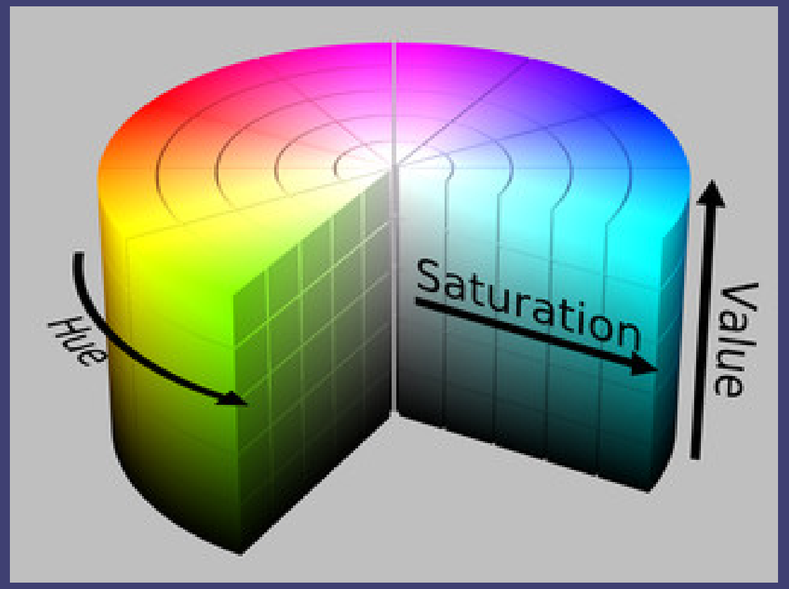
- *The image is a 3D array with the BGR values stored in each box.*
- *We will look more how they are stored when we will start Image Enhancement Program*

```
Product C:
[[[ 57  21  48   0]
  [ 71  27  60   0]
  [111  51  96   0]]

 [[ 66  16  36  18]
  [ 95  22  58  29]
  [ 83  20  46  23]]

 [[ 44  34  72  44]
  [ 34  32  72  34]
  [ 13  25  64  13]]], shape=(3, 3, 4)
```

WHAT IS BGR, RGB, HSV, GRAYSCALE ?







PROBLEM WITH BGR !!

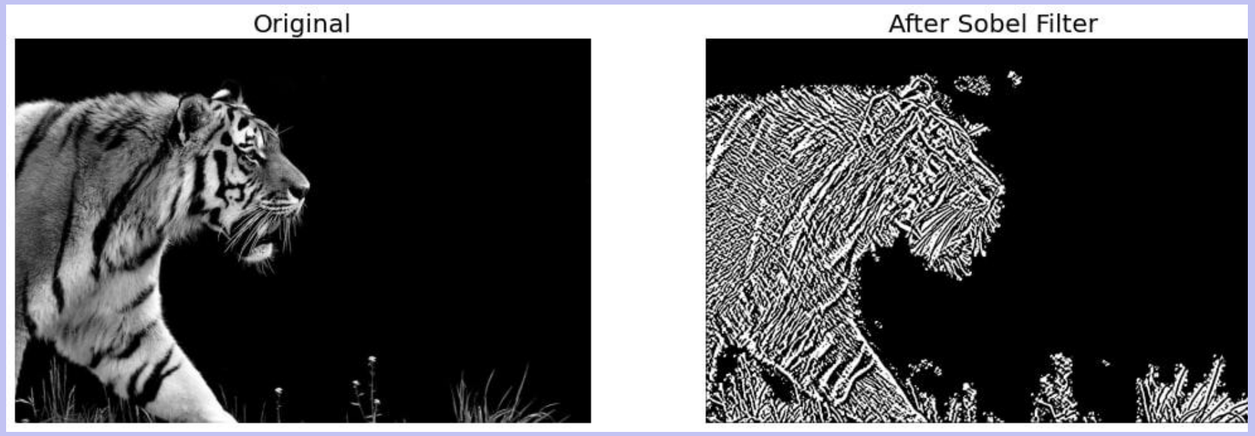
- Can not perceive its RGB value with naked eyes.
- RGB does not represent how we humans perceive color.

GUESS THE R VALUE IN THIS COLOR ?

DEMO

LETS US KNOW THE DIFFERENT FUNCTIONS YOU WERE USING!!

Canny



Sobel

Canny Edge detection is an Algorithm consisting of 4 major steps:

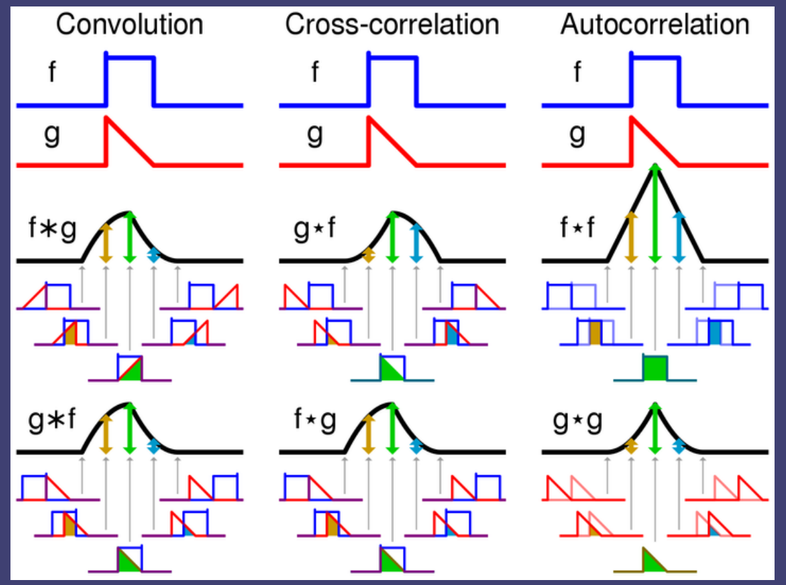
- Reduce Noise using Gaussian Smoothing.
- Compute image gradient using Sobel filter.
- Apply Non-Max Suppression or NMS to just jeep the local maxima
- Finally, apply Hysteresis thresholding which that 2 threshold values T_upper and T_lower which is used in the Canny() function.

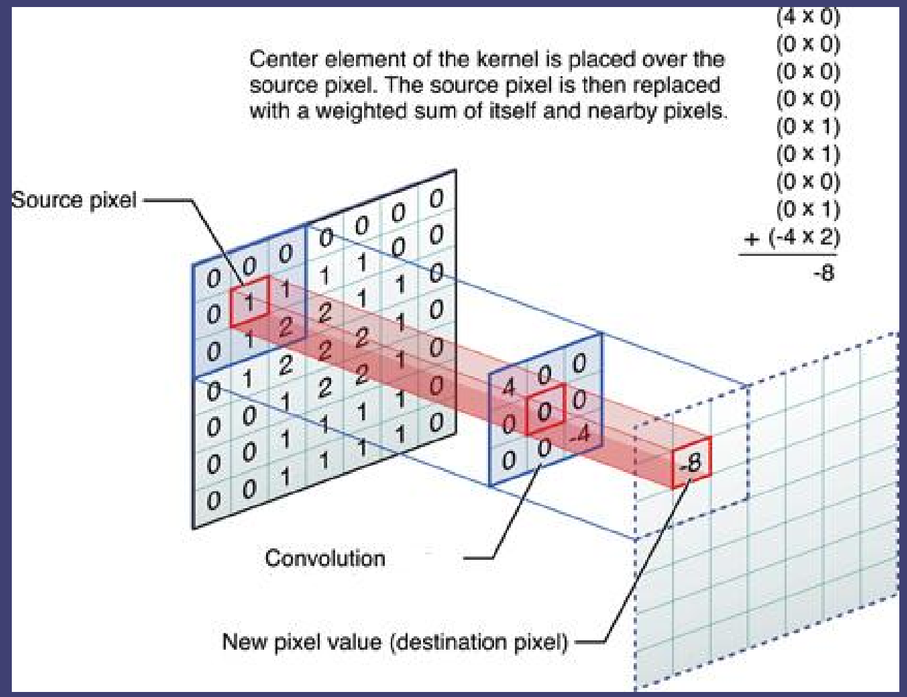
LETS START FROM BASICS !!

- *What is Convolution ??*

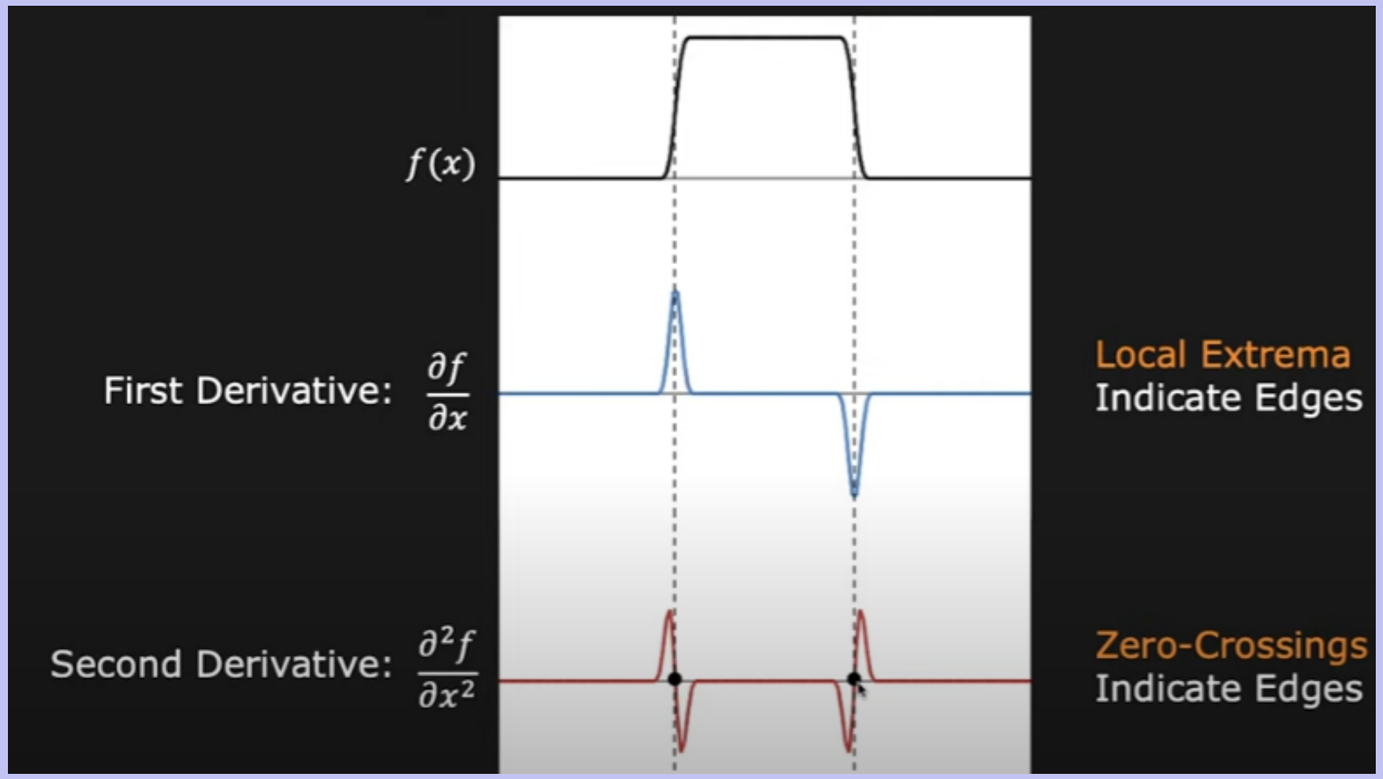
$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau$$

To convolve a kernel with an input signal:
flip the signal, move to the desired time,
and accumulate every interaction with the kernel





GRADIENT


$f(x)$
First Derivative: $\frac{\partial f}{\partial x}$
Second Derivative: $\frac{\partial^2 f}{\partial x^2}$
Local Extrema
Indicate Edges
Zero-Crossings
Indicate Edges

APPROXIMATIONS AND FILTERS

Finite difference approximations:

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right)$$

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right)$$

$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i,j}$	$I_{i+1,j}$

ε

Can be implemented as Convolution!

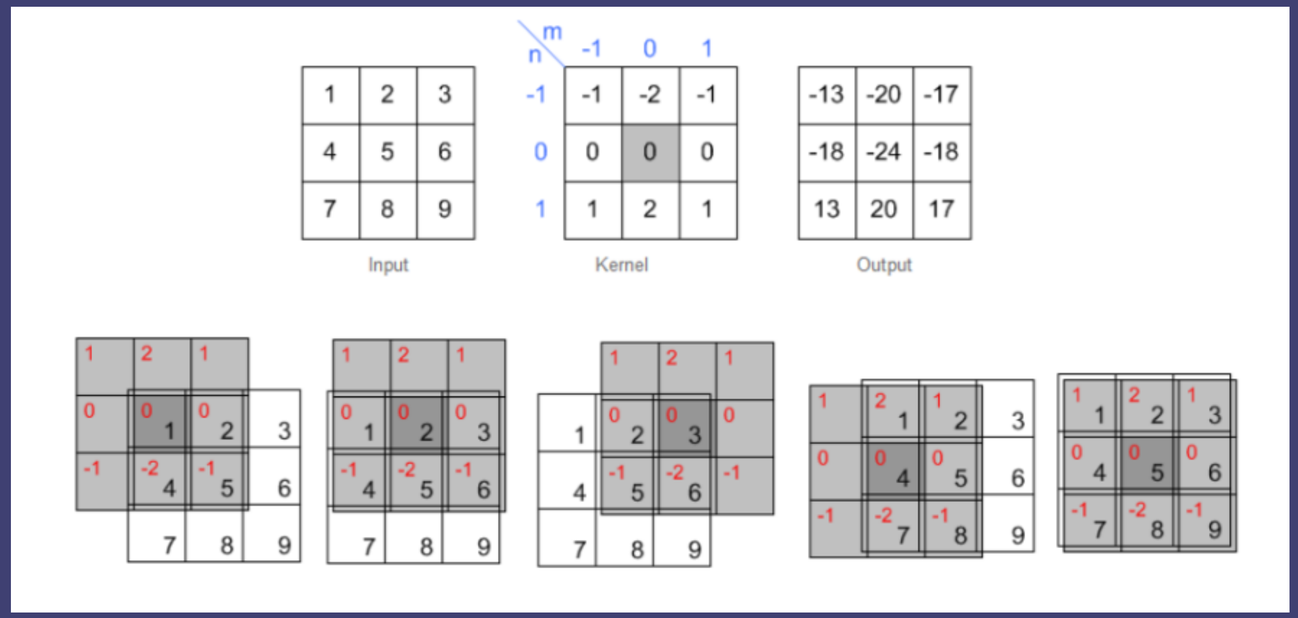
$$\frac{\partial}{\partial x} \approx \frac{1}{2\varepsilon}$$

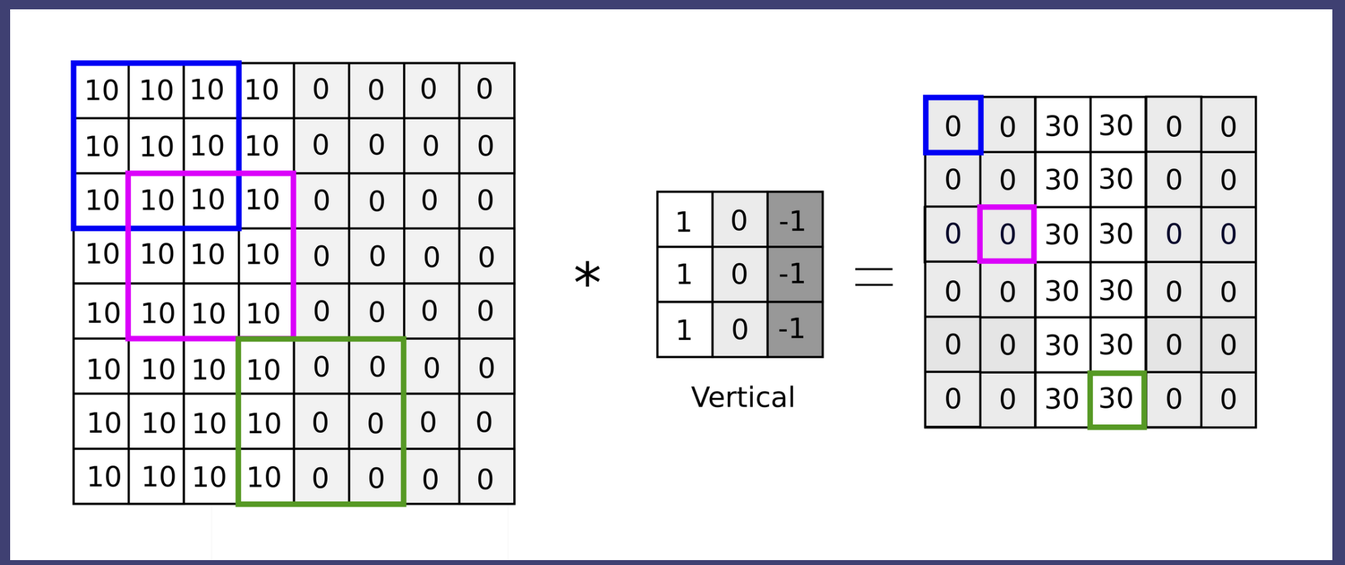
−1	1
−1	1

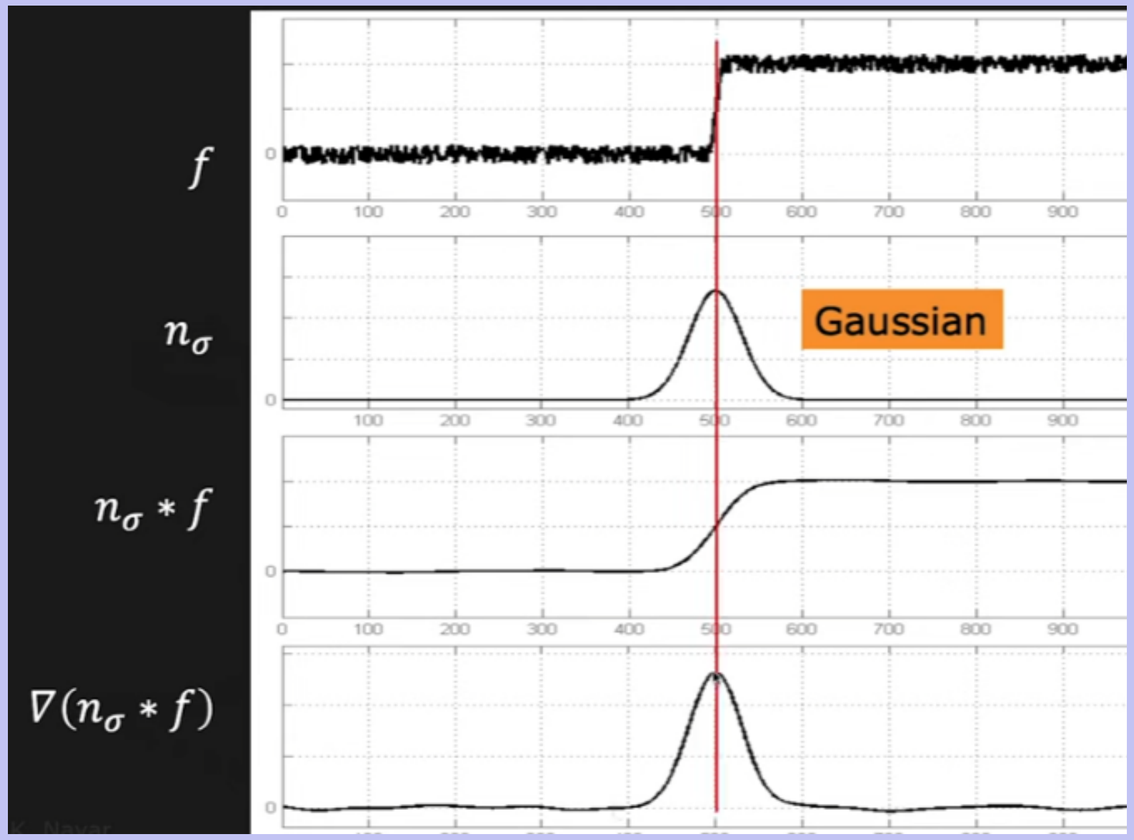
$$\frac{\partial}{\partial y} \approx \frac{1}{2\varepsilon}$$

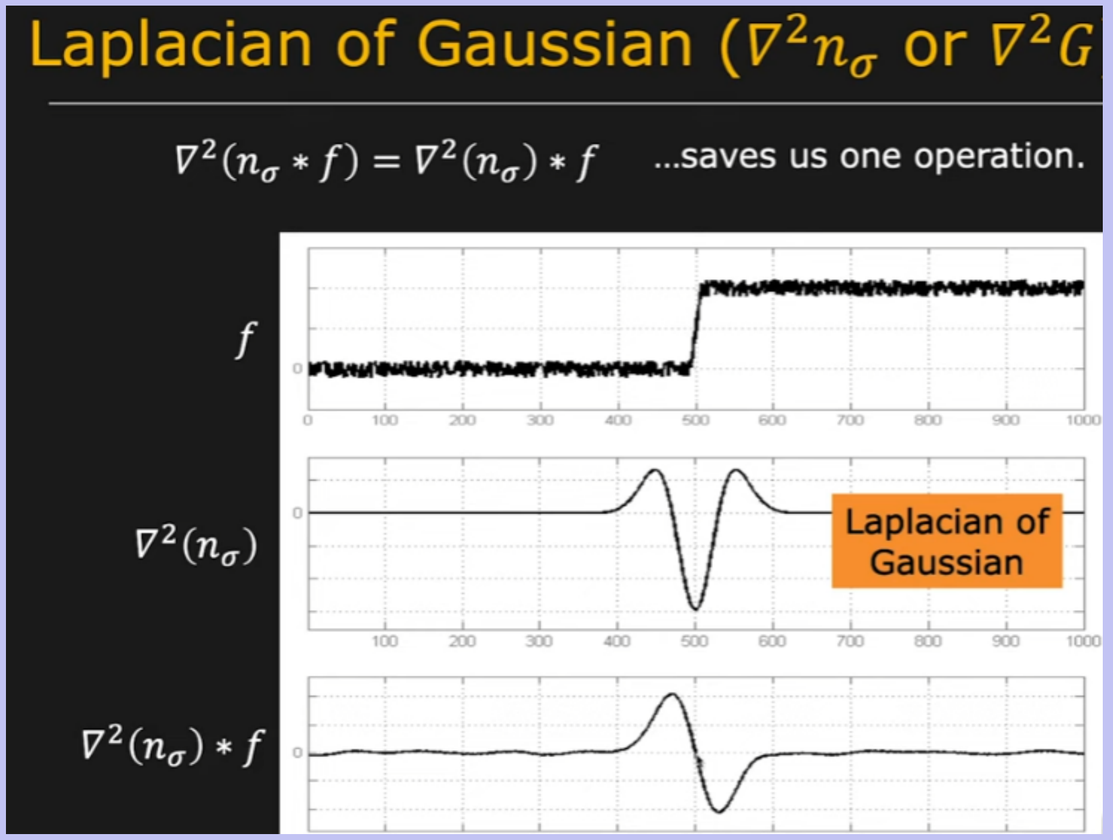
1	1
−1	−1

Gradient	Roberts	Prewitt	Sobel (3x3)	Sobel (5x5)
$\frac{\partial I}{\partial x}$	0 1 -1 0	-1 0 1 -1 0 1 -1 0 1	-1 0 1 -2 0 2 -1 0 1	-1 -2 0 2 1 -2 -3 0 3 2 -3 -5 0 5 3 -2 -3 0 3 2 -1 -2 0 2 1
$\frac{\partial I}{\partial y}$	1 0 0 -1	1 1 1 0 0 0 -1 -1 -1	1 2 1 0 0 0 -1 -2 -1	1 2 3 2 1 2 3 5 3 2 0 0 0 0 0 -2 -3 -5 -3 -2 -1 -2 -3 -2 -1

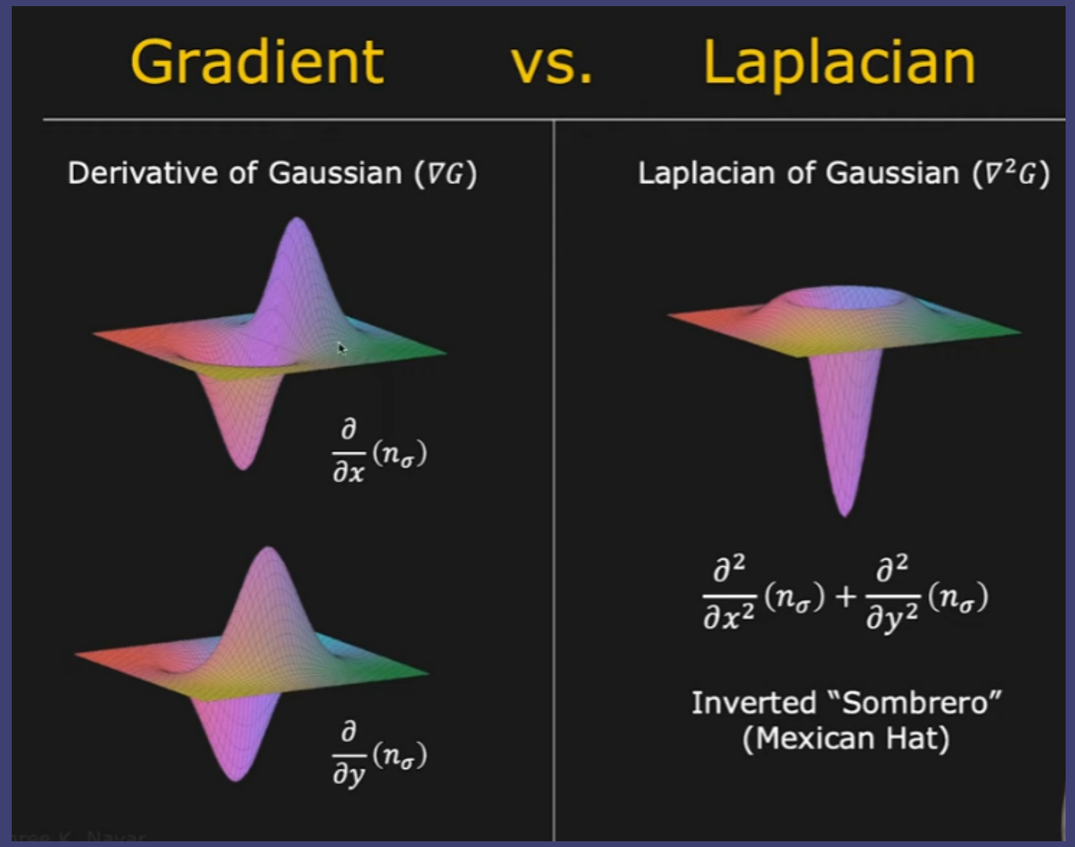

m
n
-1 0 1
-1 -1 -2 -1
0 0 0 0
1 1 2 1
1 2 3
4 5 6
7 8 9
-13 -20 -17
-18 -24 -18
13 20 17
Input
Kernel
Output


10 10 10 10 0 0 0 0
10 10 10 10 0 0 0 0
10 10 10 10 0 0 0 0
10 10 10 10 0 0 0 0
10 10 10 10 0 0 0 0
10 10 10 10 0 0 0 0
10 10 10 10 0 0 0 0
10 10 10 10 0 0 0 0
*
1 0 -1
1 0 -1
1 0 -1
Vertical
=
0 0 30 30 0 0
0 0 30 30 0 0
0 0 30 30 0 0
0 0 30 30 0 0
0 0 30 30 0 0
0 0 30 30 0 0


f
n_σ
Gaussian
$n_\sigma * f$
$\nabla(n_\sigma * f)$


Laplacian of Gaussian ($\nabla^2 n_\sigma$ or $\nabla^2 G$
$\nabla^2(n_\sigma * f) = \nabla^2(n_\sigma) * f$ ...saves us one operation.
f
$\nabla^2(n_\sigma)$
Laplacian of Gaussian
$\nabla^2(n_\sigma) * f$

EXTENDING THE CONCEPTS TO 3D FUNCTIONS



GAUSSIAN FUNCTION

Derived from 2D gaussian function

$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{\left(-\frac{u^2+v^2}{\sigma^2}\right)}$$

Example: This kernel is approximation of gaussian function

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

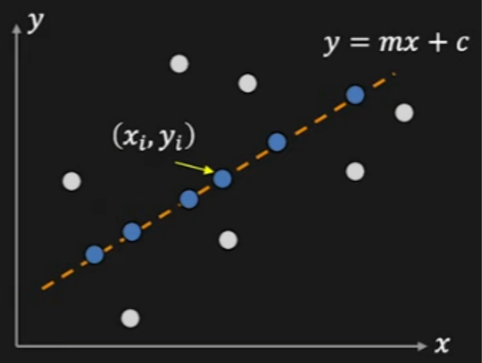
CODING TIME!

HOUGH TRANSFORM

Given: Edge Points (x_i, y_i)

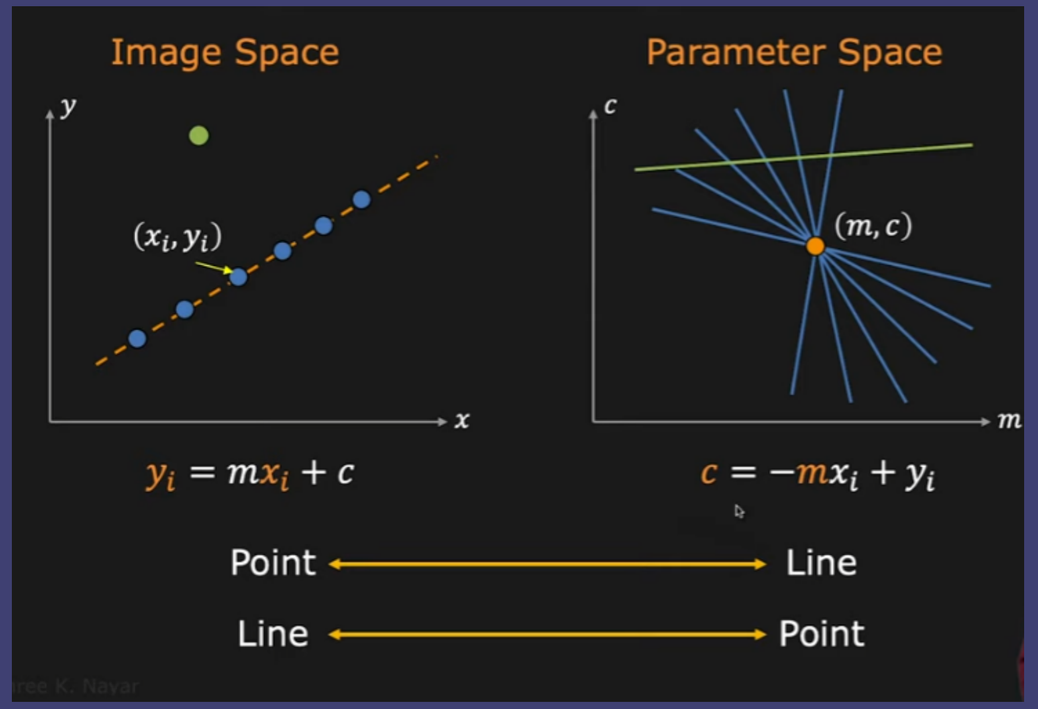
Task: Detect line

$y = mx + c$



Consider point (x_i, y_i)

$$y_i = mx_i + c \iff c = -mx_i + y_i$$



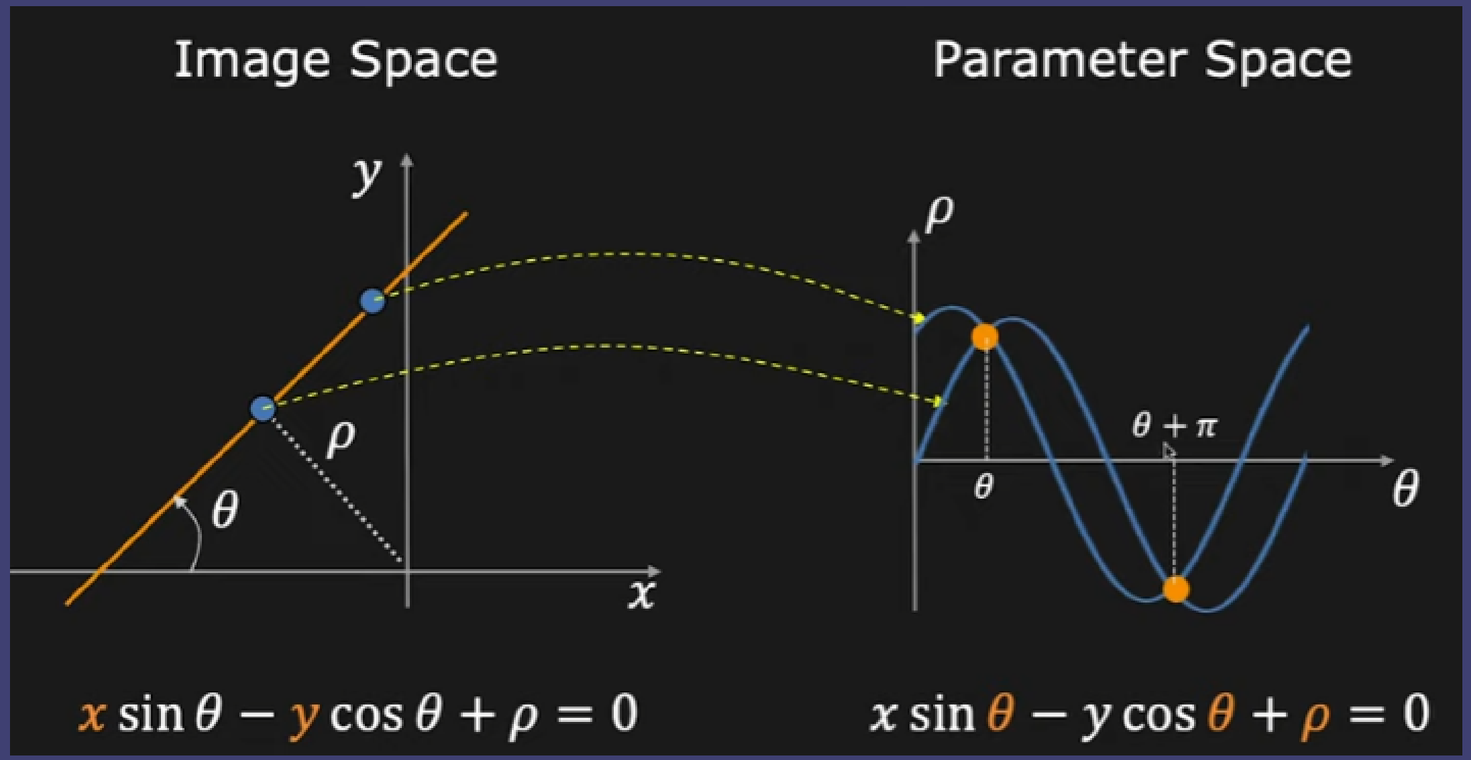
PROBLEMS WITH OUR CURRENT APPROACH :-

- *Slope of the line can with from $-\infty < m < \infty$*
- *Heavy computation*
- *More memory*

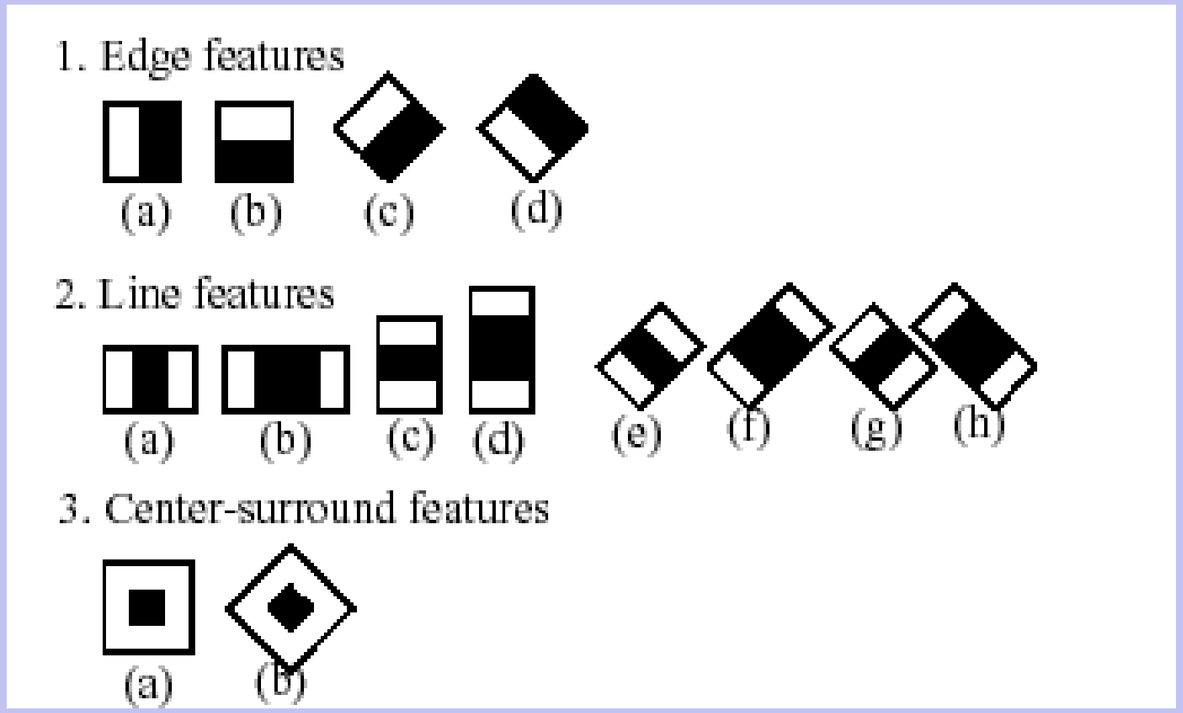
SOLUTION:-

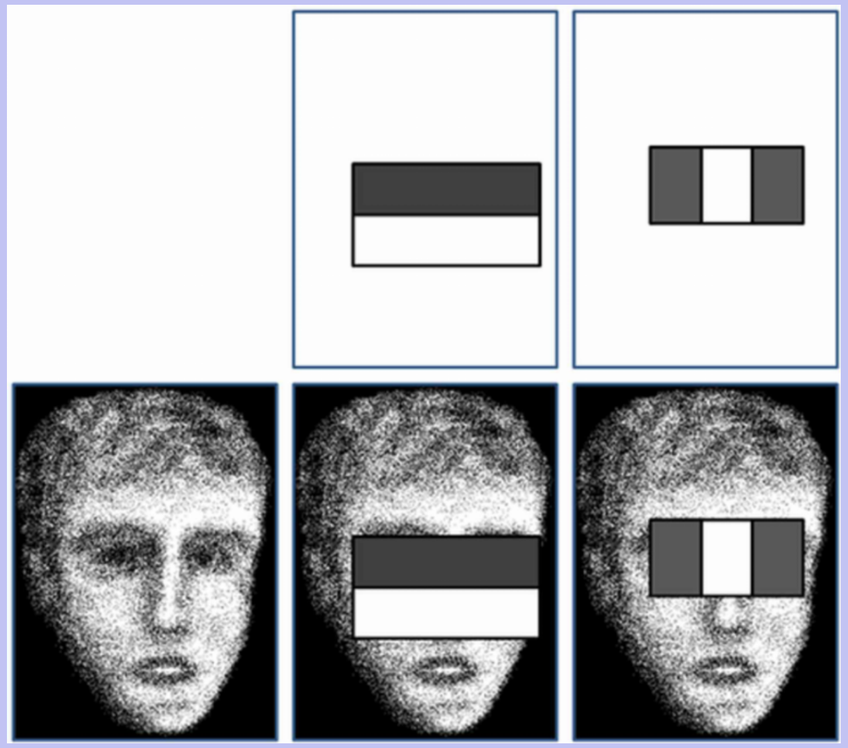
- *Use parameterization $x\sin\theta - y\cos\theta + \rho = 0$*
- *Both orientation and distance are finite*

PARAMETRIC APPROACH:-

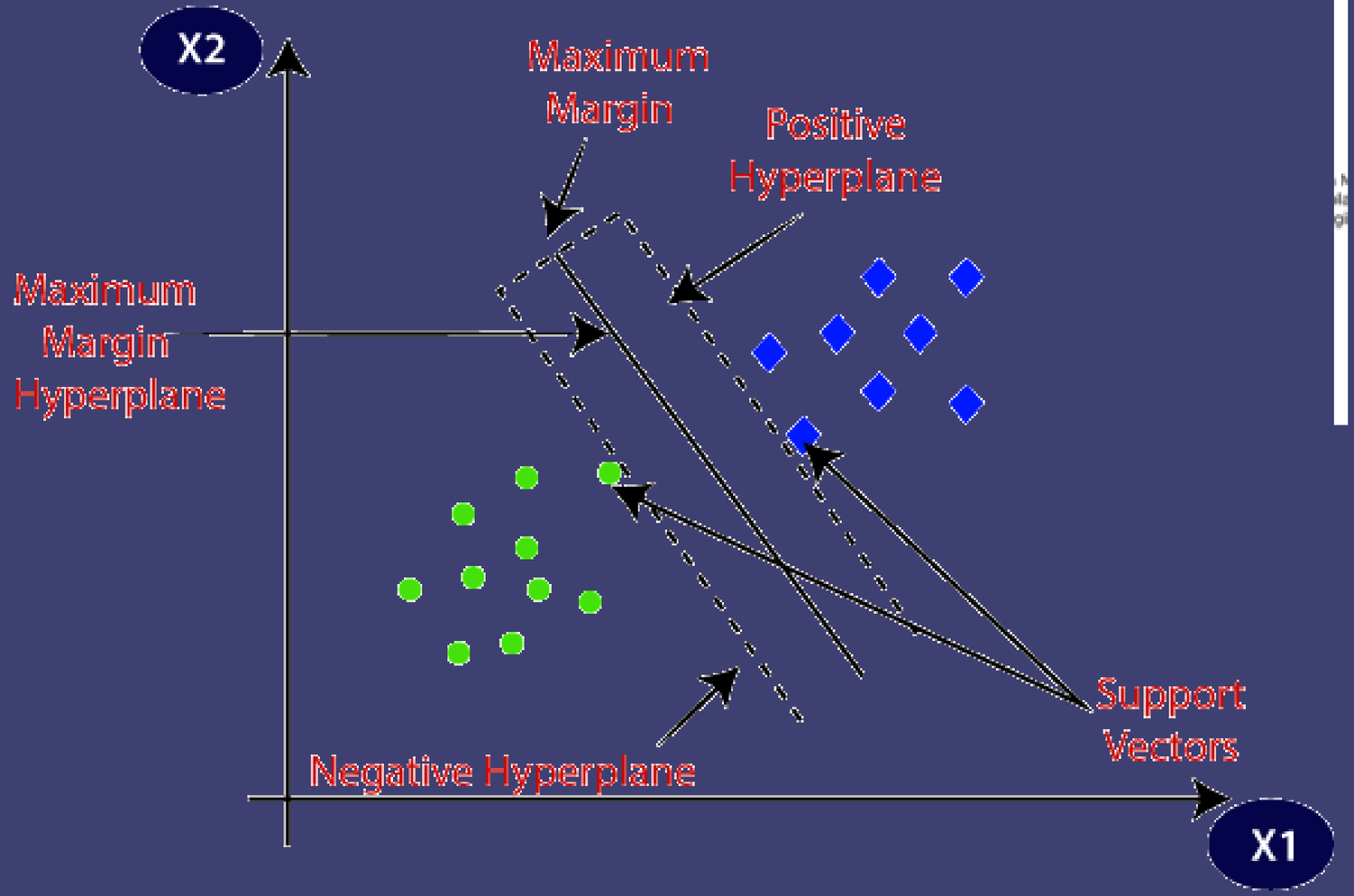


FACE DETECTION USING HAAR FEATURES





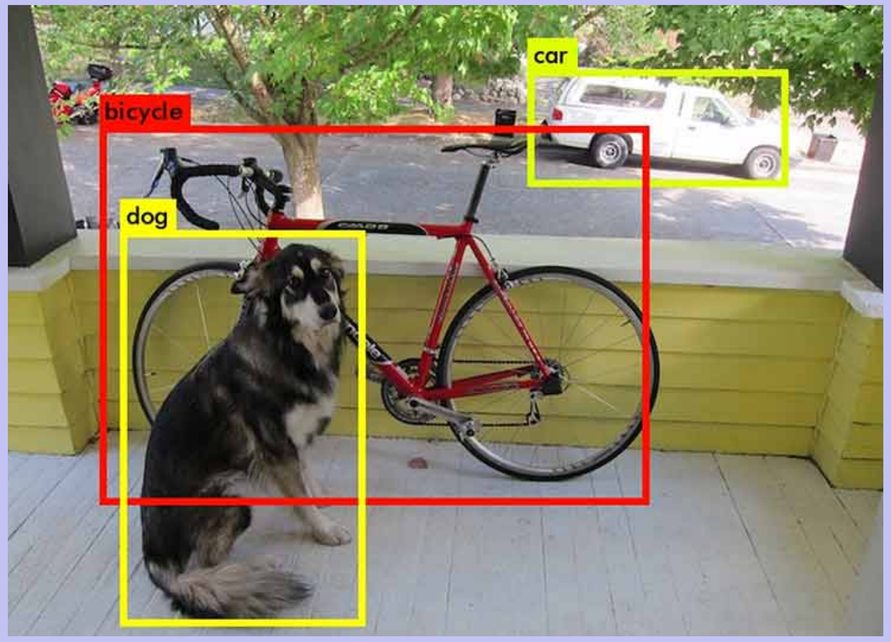
SUPPORT VECTOR MACHINE



MASKING

YOLO - YOU ONLY LOOK ONCE

Yolo is state-of-the-Art, real-time object detection system



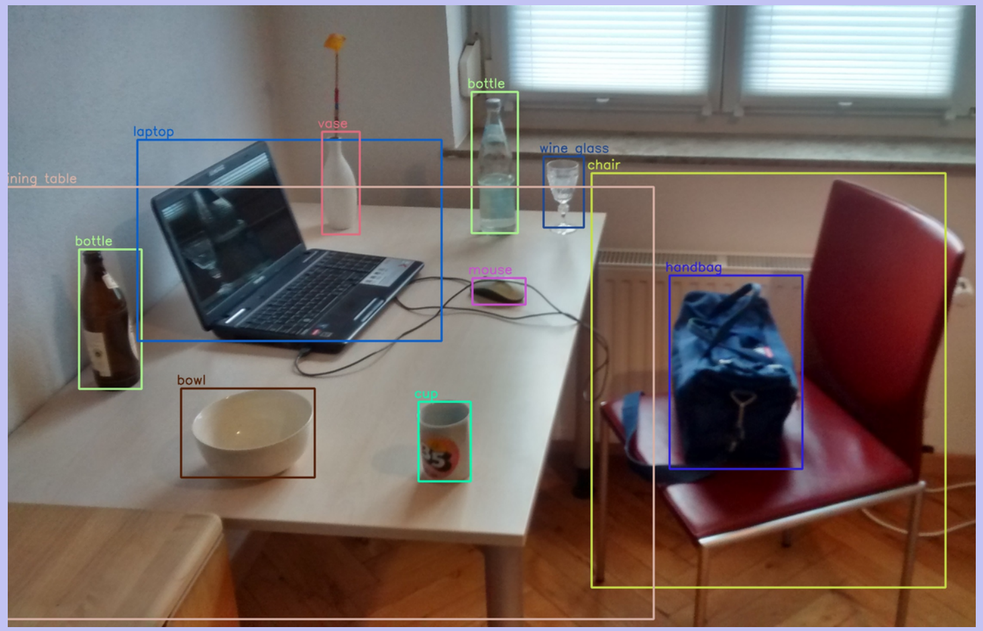
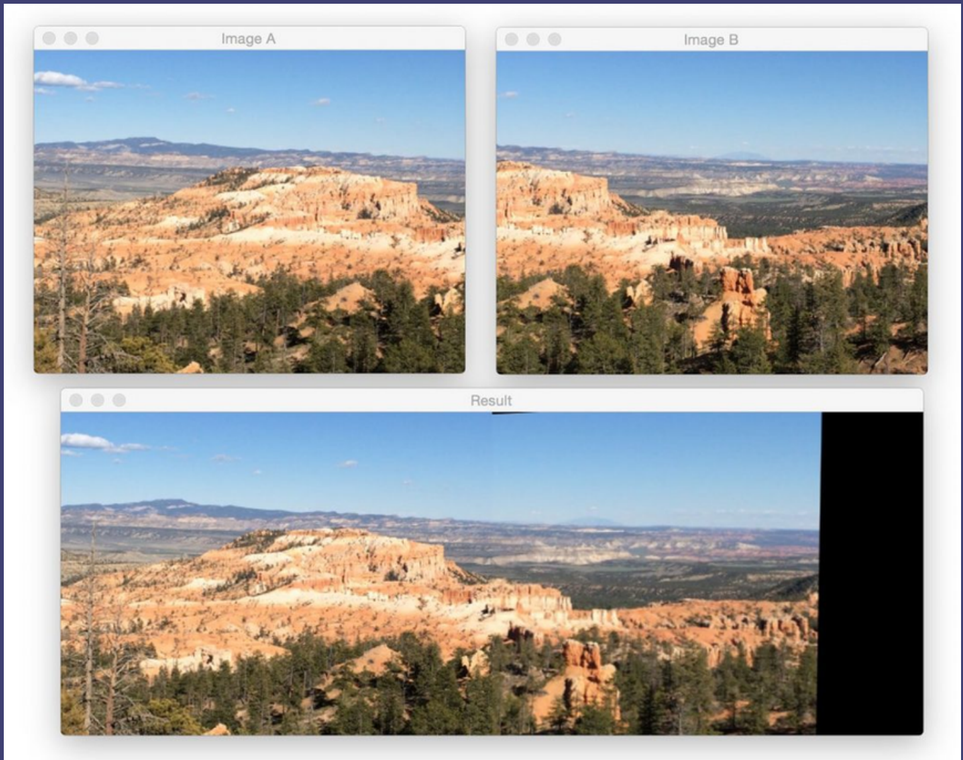
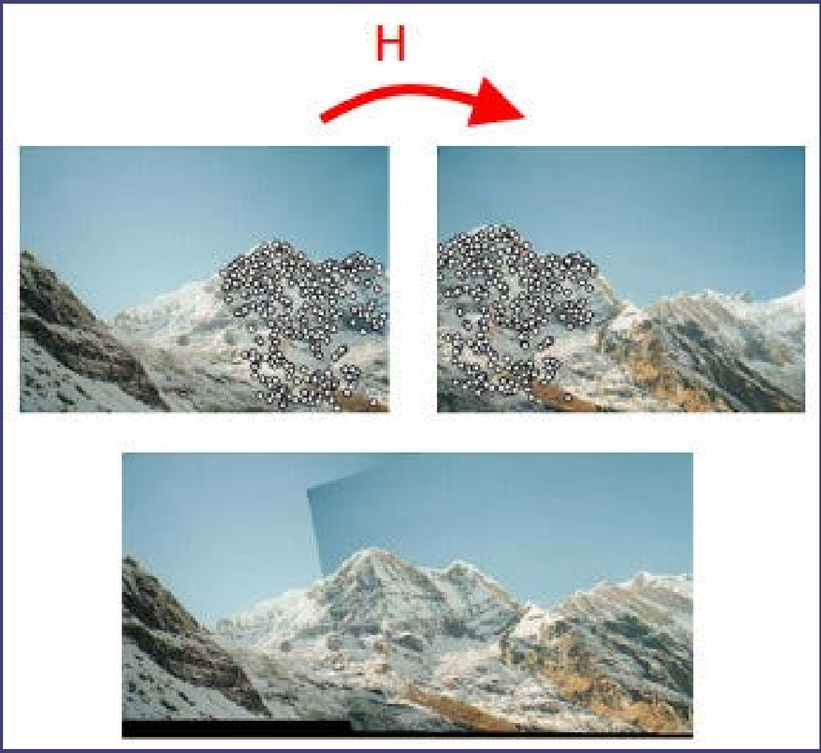


IMAGE STITCHING

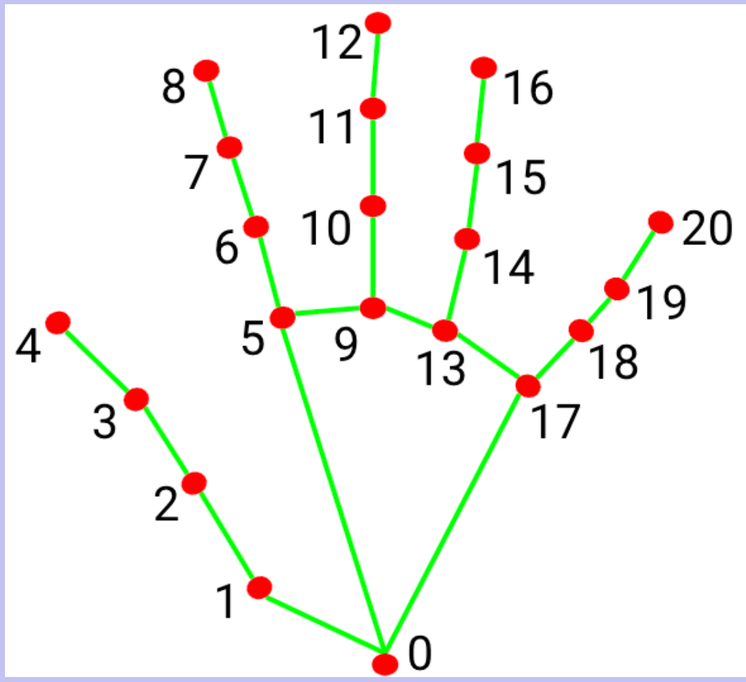
*Image stitching or photo stitching is **the process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image***





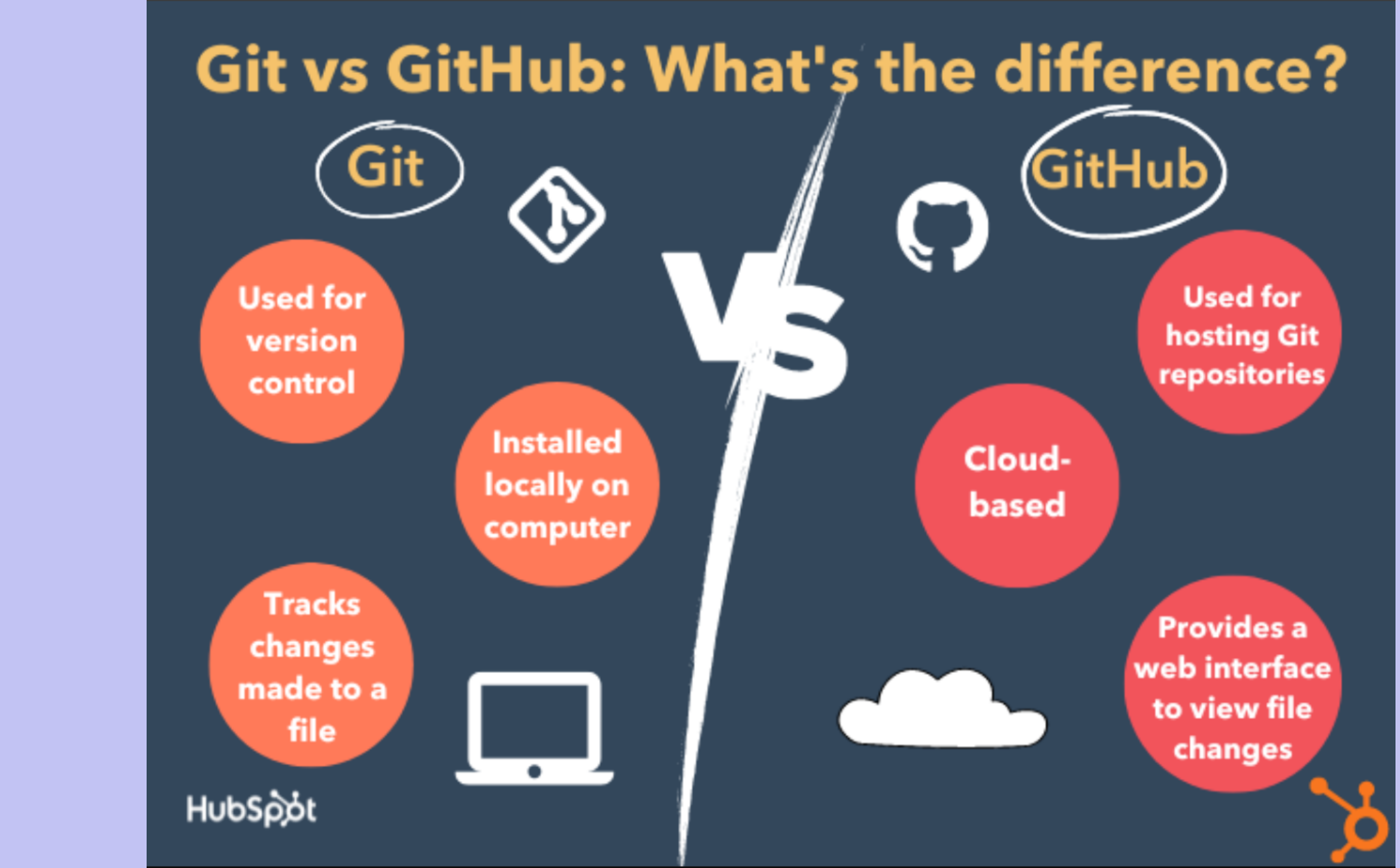
HAND GESTURE RECOGNITION USING MEDIAPIPE

MediaPipe



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Git vs GitHub: What's the difference?



Thank you!