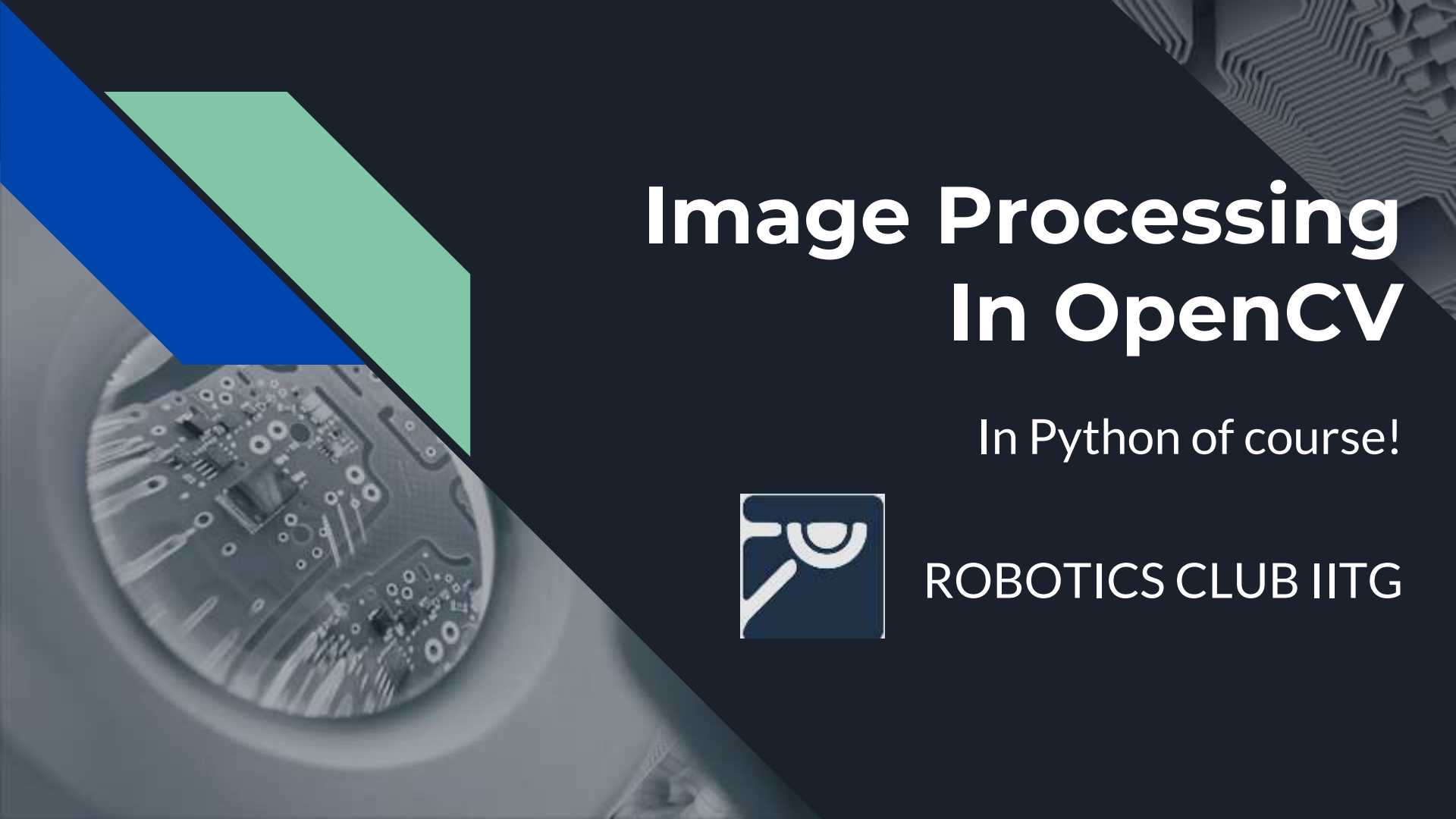


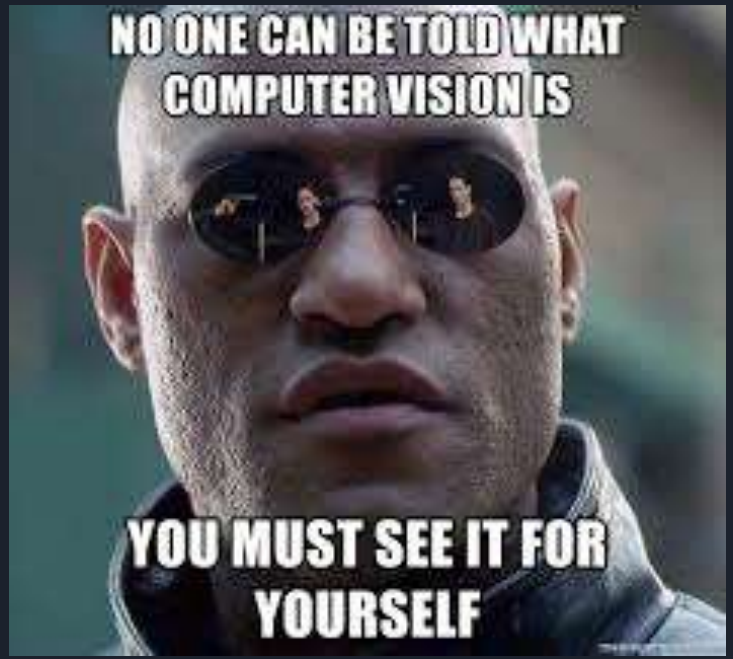
Image Processing In OpenCV

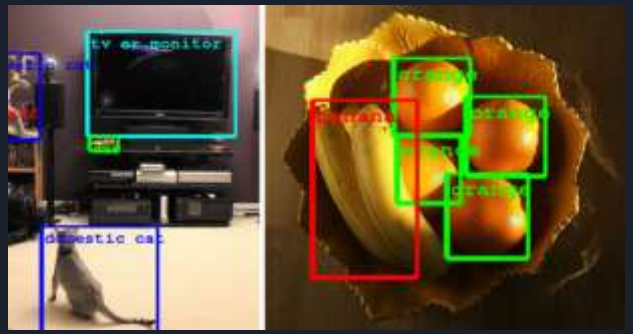
In Python of course!

ROBOTICS CLUB IITG

What is Computer Vision?

“Computer Vision is an interdisciplinary scientific field that deals with how computers can be made to gain high-level understanding from digital images or videos.” -Wikipedia





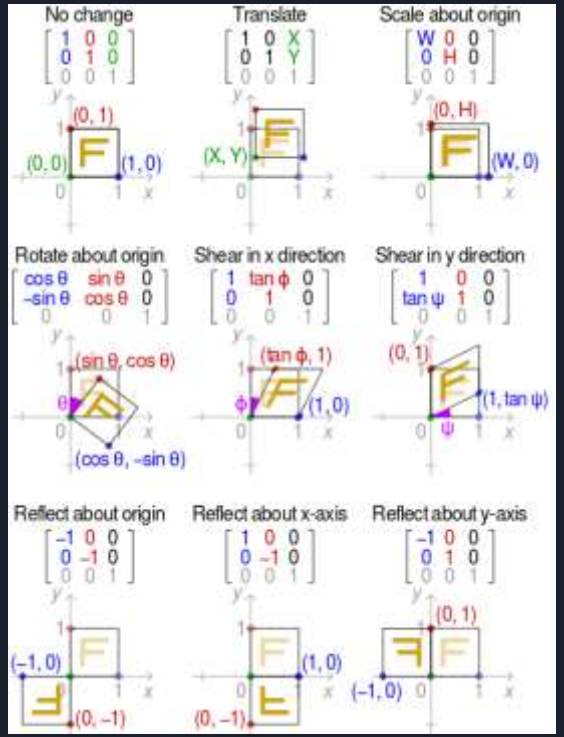
Object Recognition

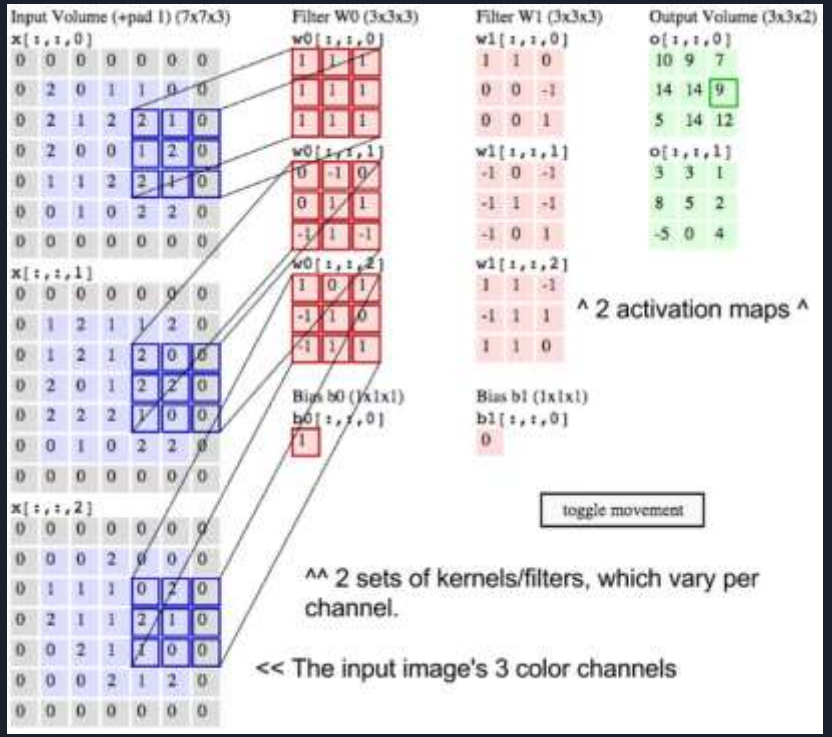
Face Recognition

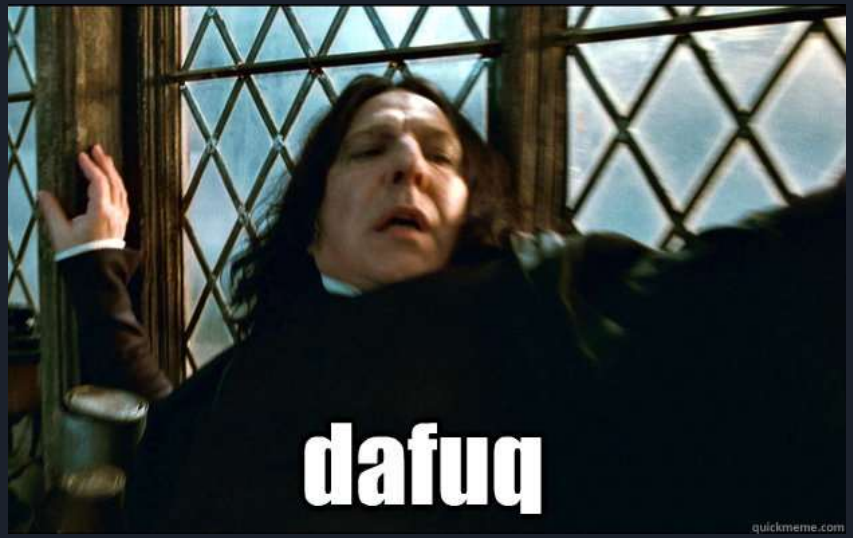
Pose Estimation

What is Image Processing then?

“A method to perform some operations on an image to extract useful information from it”






dafuq
quickmeme.com

Seems Daunting?

We will break all of it down for you!
Using OpenCV ofcourse!

OpenCV is a library which provides functions to perform all of these complex operations on images!

What is an image ?

There are two types of recording images:

1. Analog Image : Analog images are the type of images that we, as humans, look at. They also include such things as photographs, paintings, TV images, and all of our medical images recorded on film or displayed on various display devices. What we see in an analog image is various levels of brightness (or film density) and colors. It is generally continuous and not broken into many small individual pieces.

1. Digital Image : A digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as pels, pixels, picture elements or image elements

A digital image is a numeric representation, normally binary, of a two-dimensional image.

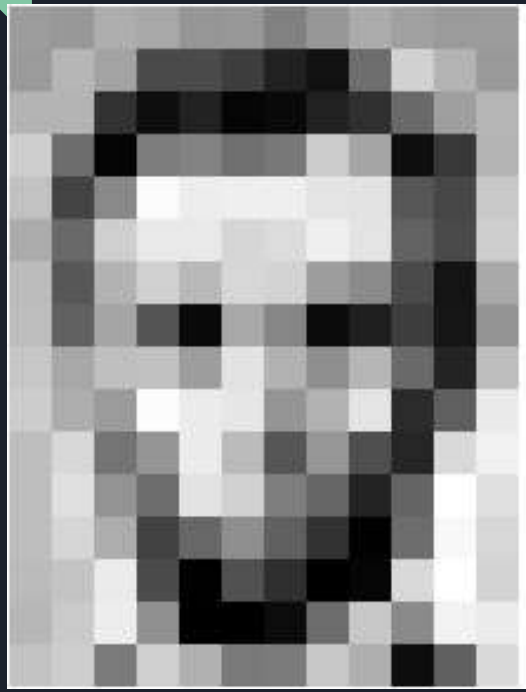
Mostly used types of images:

1. Grayscale images 2. RGB images

A Grayscale Image

A Grayscale Image

A grayscale or greyscale image is one in which the value of each pixel is a single sample representing only an amount of light, that is, it carries only intensity information. Grayscale images, a kind of black-and-white or gray monochrome, are composed exclusively of shades of gray. The contrast ranges from black at the weakest intensity to white at the strongest



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Color Image

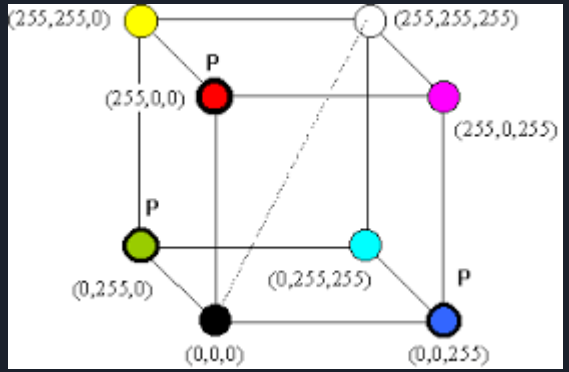
A (digital) color image is a digital image that includes color information for each pixel.

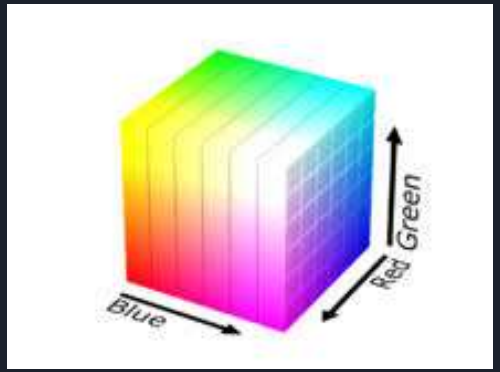
Any color image can be shown with two color models:

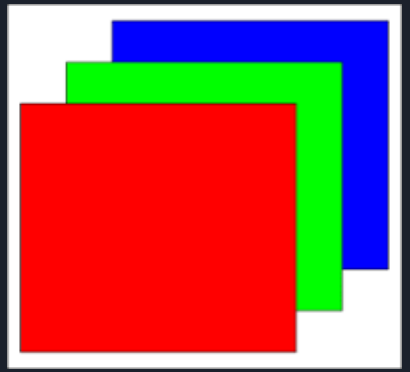
1. RGB
2. HSV

RGB color model

The **RGB color model** is an additive **color model** in which red, green and blue light are added together in various ways to reproduce a broad array of **colors**.

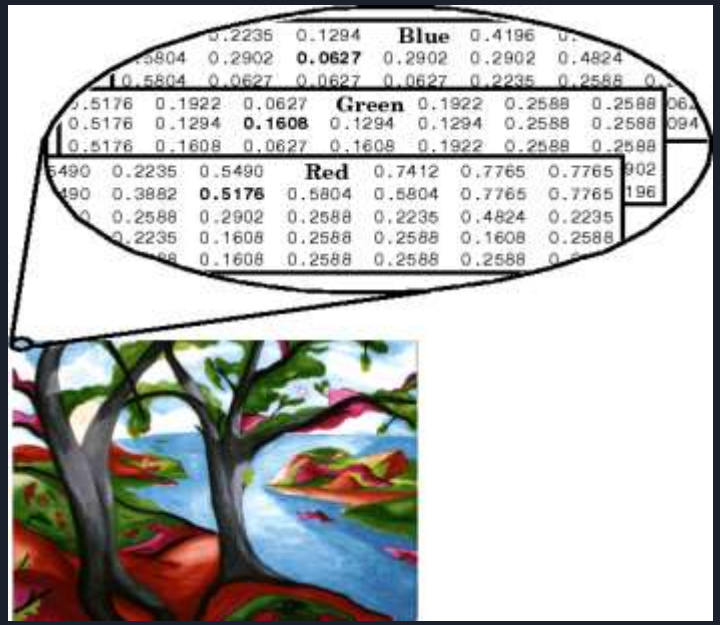






RGB color model

It consists of three matrices consisting of intensity of three primary colors i.e. red, blue and green for a particular pixel.



HSV Color Model

The HSV color wheel is sometimes depicted as a cone or cylinder, but always with these three components **HSV** (hue, saturation, value):

Hue :

Hue is the color portion of the color model, expressed as a number from 0 to 360 degrees:

Saturation :

Saturation is the amount of gray in the color, from 0 to 100 percent. Reducing the saturation toward zero to introduce more gray produces a faded effect. Sometimes, saturation is expressed in a range from just 0–1, where 0 is gray and 1 is a primary color.

Value (or Brightness) :

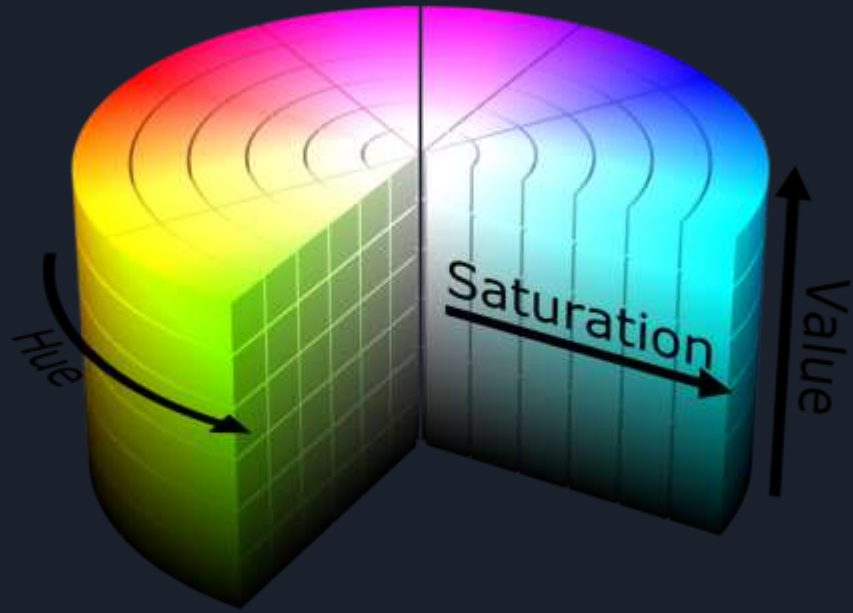
Value works in conjunction with saturation and describes the brightness or intensity of the color, from 0–100 percent, where 0 is completely black, and 100 is the brightest and reveals the most color.

HSV Color Model

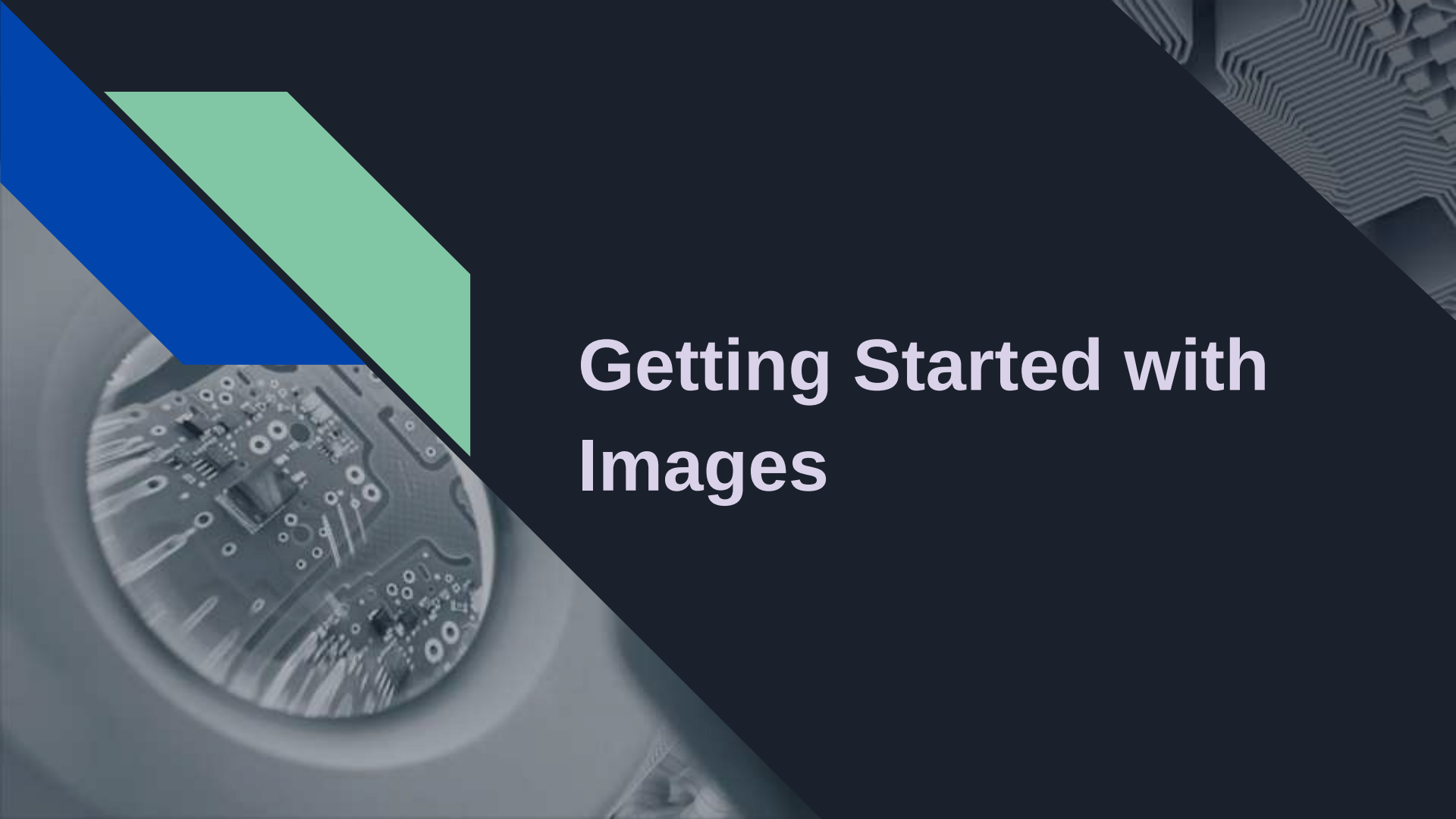
Hue

* In HSV, hue represents color. In this model, hue is an angle from 0 degrees to 360 degrees.

Angle	Color
0-60	Red
60-120	Yellow
120-180	Green
180-240	Cyan
240-300	Blue
300-360	Magenta



ne
ts
gital



Getting Started with Images

Important Functions

cv2.imread() → used to read an image

- 1st argument is name of image and 2nd argument is a flag .
- Example - cv2.imread('messi5.jpg',0)

cv2.imshow() → it is used to display an image

- 1st argument is window name and 2nd argument is name of image.
- Example - cv2.imshow('image',img)

cv2.imwrite() → used to save an image.

- 1st argument is name you want to give to your image and 2nd argument is the image itself .
- Ex : cv2.imwrite('messi.png' , img)

cv2.waitKey() → is a keyboard binding function. Its argument is the time in milliseconds.

cv2.destroyAllWindows() → simply destroys all the windows we created.

Getting Started with Videos

cv2.VideoCapture() → To capture a video, you need to create a VideoCapture object.

- Its argument can be either the device index or the name of a video file.
- cv2.VideoCapture(‘car.mp4’)
  → Used to play any saved video
- cv2.VideoCapture(0)
  → used to take video from laptop webcam
- cv2.VideoCapture(1)
  → used to take video from any external camera

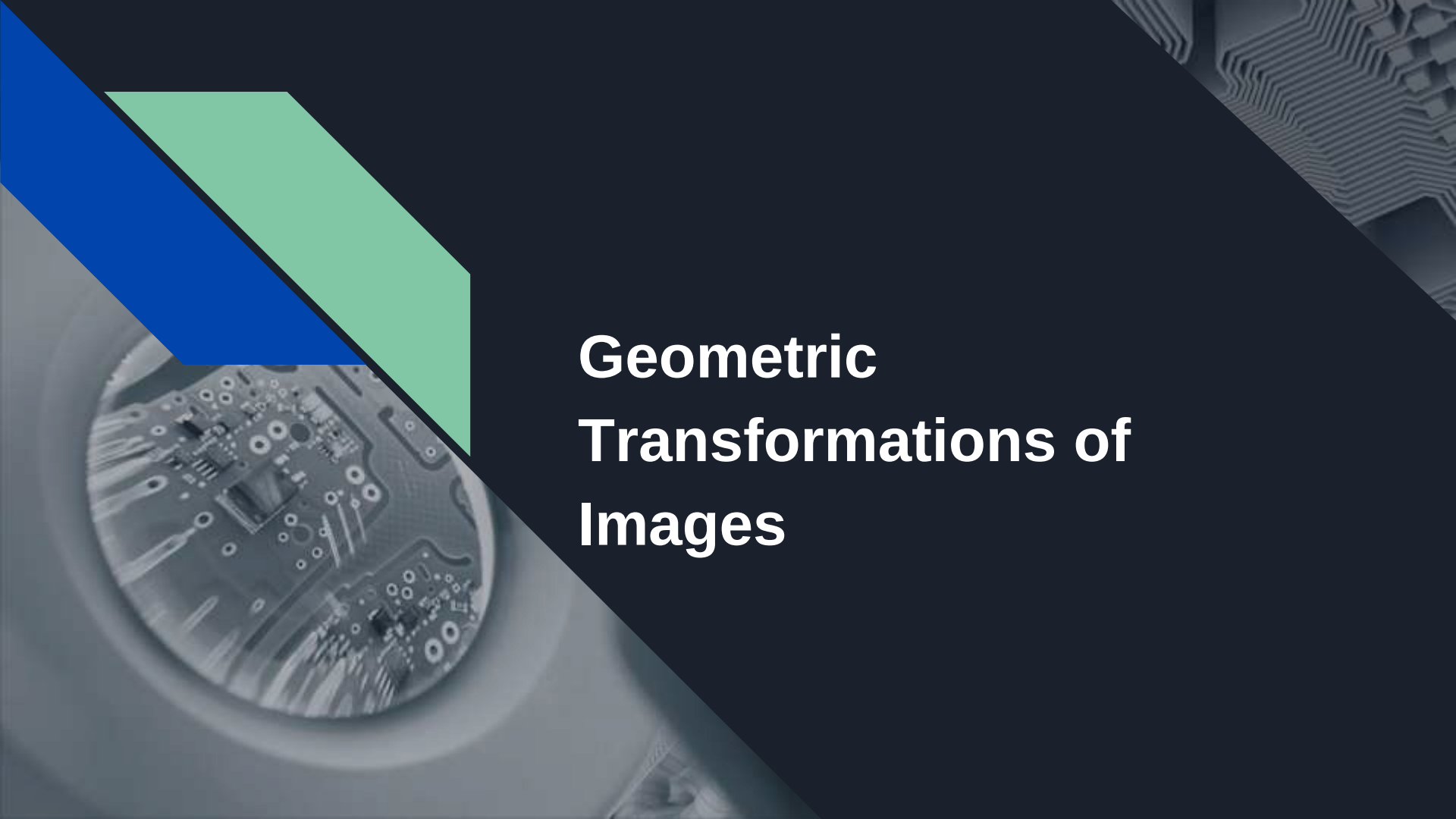
Changing Color-space

- cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)

  → used to convert BGR image to Gray Scale image

- cv2.cvtColor(input_image, cv2.COLOR_BGR2HSV)

  → used to convert BGR image to HSV image



Geometric Transformations of Images

- Learn to apply different geometric transformation to images like Scaling ,translation, rotation, affine transformation etc.

Scaling

- Scaling is just resizing of the image.
- OpenCV comes with a function **cv2.resize()** for this purpose.
- The size of the image can be specified manually, or you can specify the scaling factor.
- We use interpolation **cv2.INTER_CUBIC** (slow) method for Scaling.

Translation

- Translation is the shifting of object's location.
- If you know the shift in (x,y) direction , let it be (tx,ty) you can create the transformation matrix **M**

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

Rotation

- Rotation of an image for an angle  is achieved by the transformation matrix of the form

$$M = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- To find this transformation matrix, OpenCV provides a function, **cv2.getRotationMatrix2D**. Check below example which rotates the image by 90 degree with respect to center without any scaling.

Drawing and Writing on Image

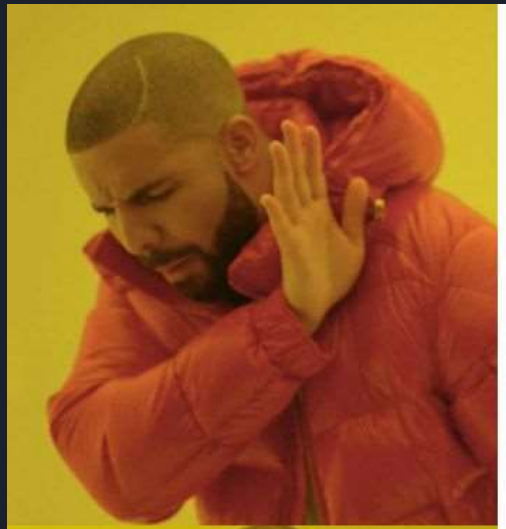
- We're going to be covering how to draw various shapes on your images and videos.
- The **cv2.line()** takes the accompanying parameters: where, begin arranges, end facilitates, shading (bgr), line thickness.
- The **cv2.rectangle** the parameters takes : upper left arranges, base right facilitate, shading, and line thickness.
- How about a circle?

- For circle we use **cv2.circle** and the parameters here are the picture/outline, the focal point of the circle, the sweep, shading, and afterwards thickness.
- Notice we have a - 1 for thickness. This implies the protest will really be filled in so we will get a filled in circle.
- CAN YOU DRAW A POLYGON ?

For writing on images use cv2.putText and need to specify

- Text data that you want to write
- Position coordinates of where you want put it (i.e. bottom-left corner where data starts).
- Font type (Check **cv2.putText()** docs for supported fonts)
- Font Scale (specifies the size of font)
- regular things like color, thickness, lineType etc. For better look, lineType = cv2.LINE_AA is recommended.

BLURRING AND FILTERING!



What you think filter is

What filter means for us

$$\frac{1}{273}$$

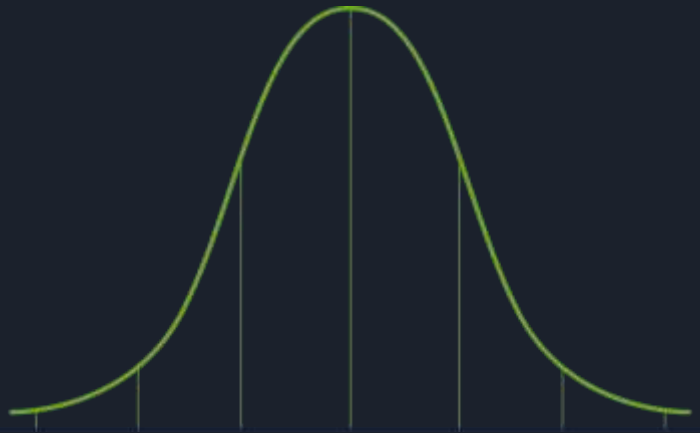
1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

GAUSSIAN BLURRING

Mathematical Definition -

$$f_g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-a)^2}{2\sigma^2}}$$

What it looks like -



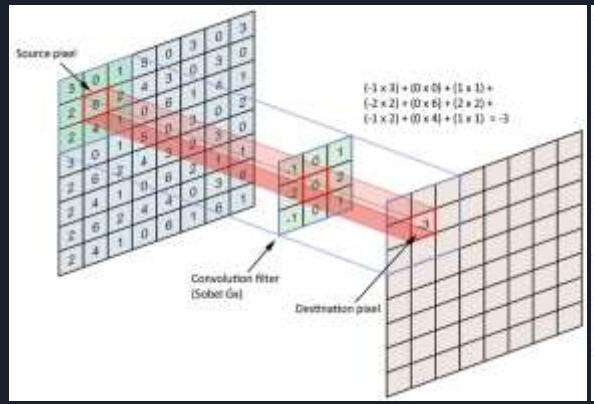
GAUSSIAN BLURRING

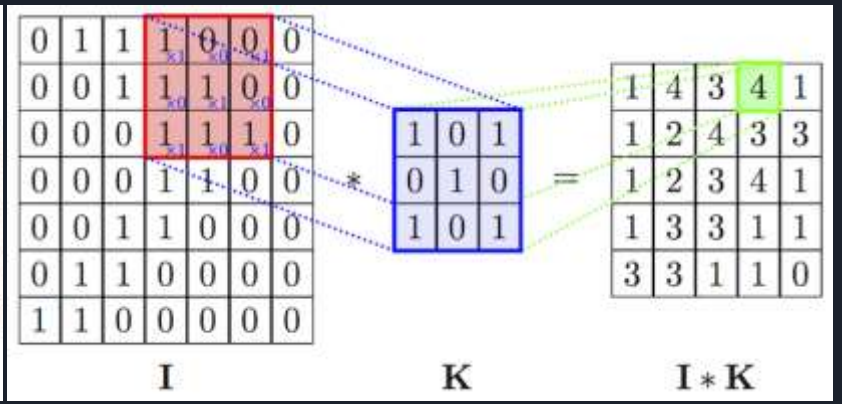
The Gaussian Kernel -

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

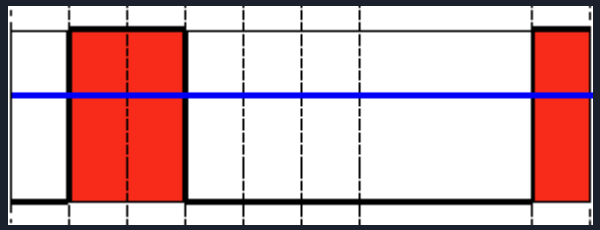
GAUSSIAN BLURRING

Convolution?

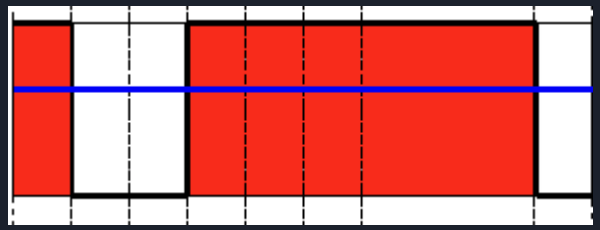




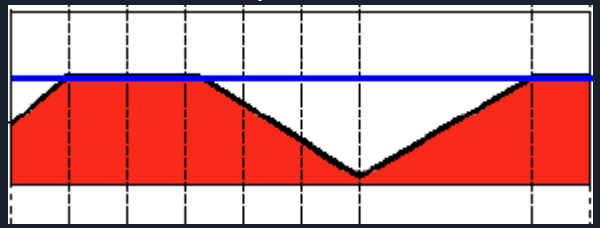
THRESHOLDING



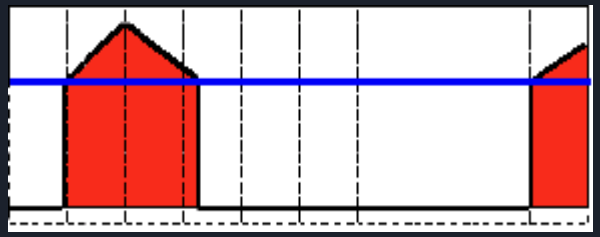
Threshold Binary



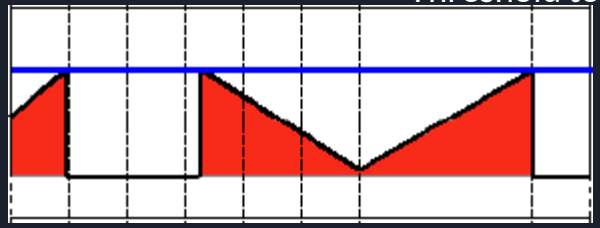
Threshold Binary Inverted



Truncate



Threshold to zero

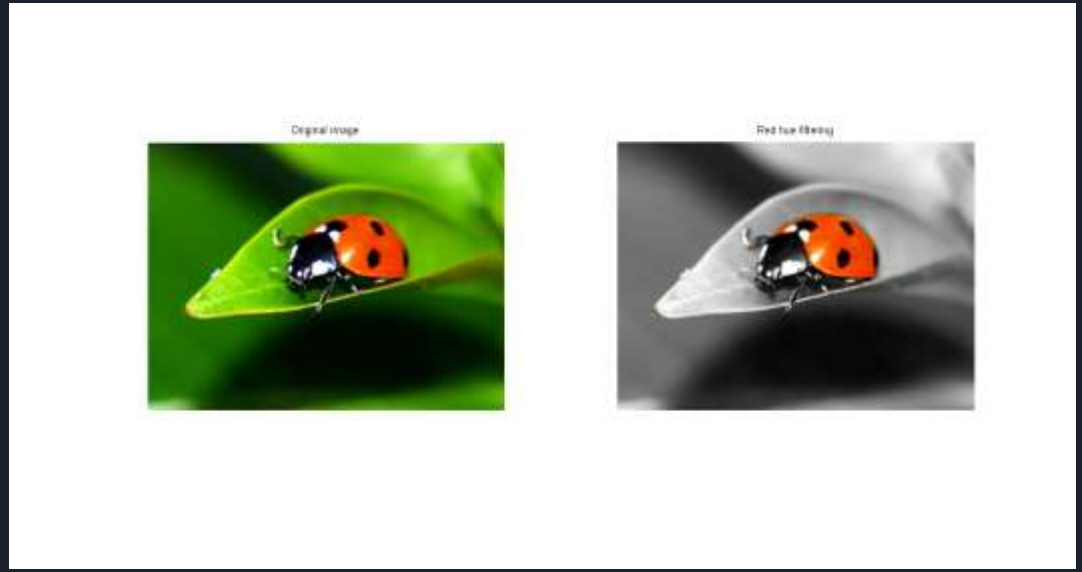


Threshold to zero inverted

FILTERS and FILTERING

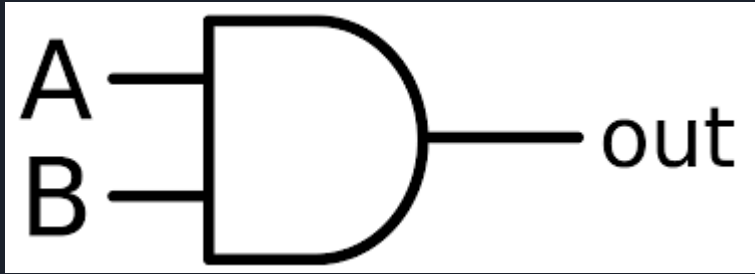
*Finally!

Basic Filtering - Color(Hue) Range Filtering



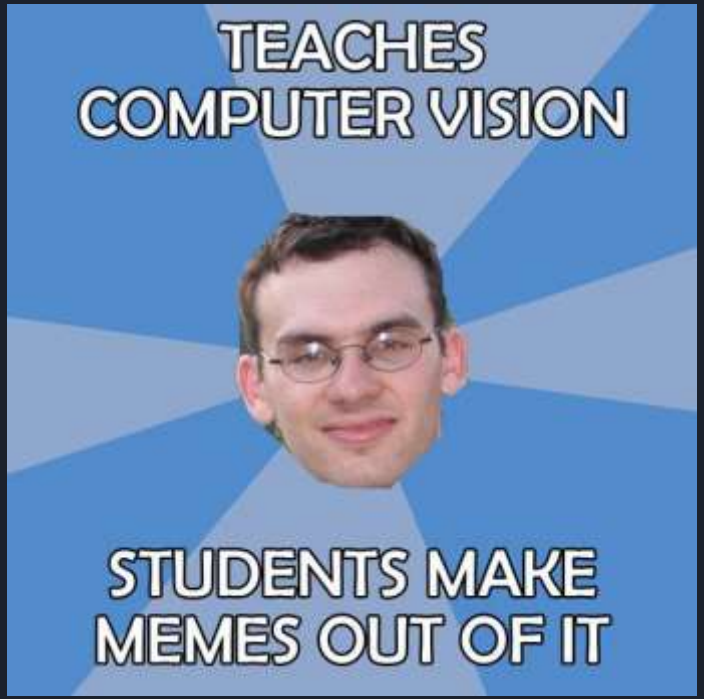
Why do all of this? Why to blur an image? Why filter the image?

BITWISE AND will explain all of it!



cv2.bitwise_and() is basically applying AND Gate on each pixel

Now what?



Canny Edge Detection

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-stage algorithm and we will go through main stages.

cv2.Canny(). We will see how to use it. First argument is our input image. Second and third arguments are our *minVal* and *maxVal* respectively. Third argument is *aperture_size*. It is the size of Sobel kernel used for find image gradients. By default it is 3. Last argument is *L2gradient* which specifies the equation for finding gradient magnitude. If it is True, it uses the equation mentioned above which is more accurate,by default, it is False.

Function: cv2.Canny(img,minval,maxval)

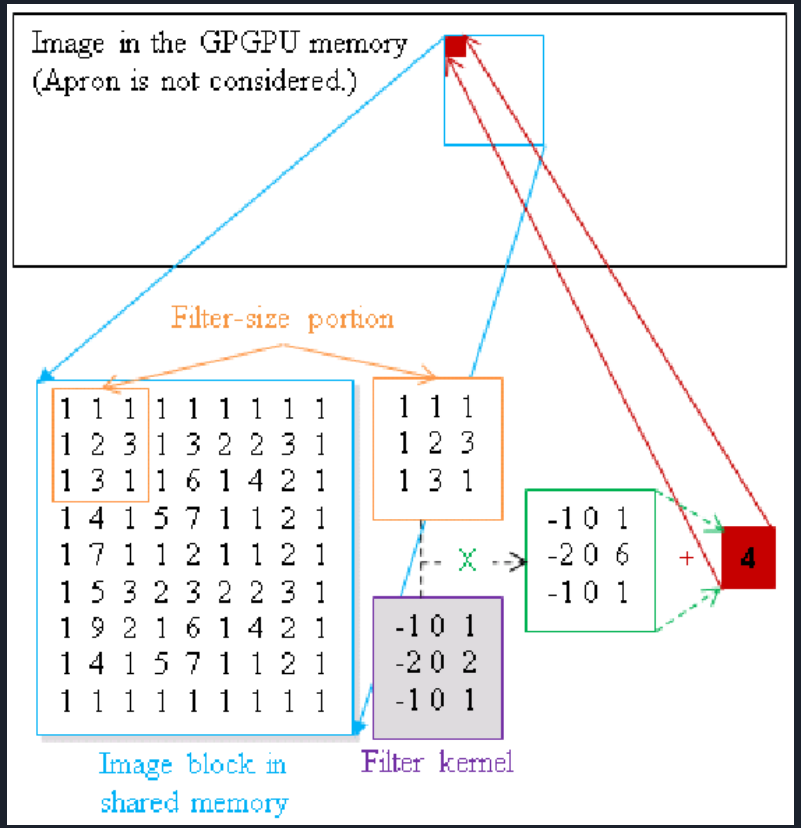


Image in the GPGPU memory
(Apron is not considered.)
Filter-size portion
1 1 1 1 1 1 1 1
1 2 3 1 3 2 2 3 1
1 3 1 1 6 1 4 2 1
1 4 1 5 7 1 1 2 1
1 7 1 1 2 1 1 2 1
1 5 3 2 3 2 2 3 1
1 9 2 1 6 1 4 2 1
1 4 1 5 7 1 1 2 1
1 1 1 1 1 1 1 1 1
1 1 1
1 2 3
1 3 1
X
-1 0 1
-2 0 2
-1 0 1
-1 0 1
-2 0 6
-1 0 1
+
4
Image block in shared memory
Filter kernel

Code for implementing the canny edge detection

```
import cv2
import numpy as np
cap = cv2.VideoCapture(0)
while True :
        ret, img = cap.read()
        cimg = cv2.Canny(img,100,200)
        cv2.imshow('45',cimg)
        cv2.imshow('45-color',img)
        if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cv2.waitKey(0)
cap.release()
cv2.destroyAllWindows()
```