



Why Computer Vision



Uses in Sports

- Player Pose Detection: AI vision can be used to recognize patterns between human body movement and pose over multiple frames in video footage or real-time video streams.
- Near Real Time Coaching: CAN help in reducing feedback time
- Ball Tracking: Tracks a ball which can further be used to develop a cricket playing bot.
- Self-training FeedBack





Traffic Detection



Used in Self Driving Cars





Tesla AutoPilot





Digital Image





IMage Representation

Images in computer system are made up of thousands of small coloured dots, Known as pixels
(short for picture elements)
They are stored as a array(3d or 2d) in our computers.

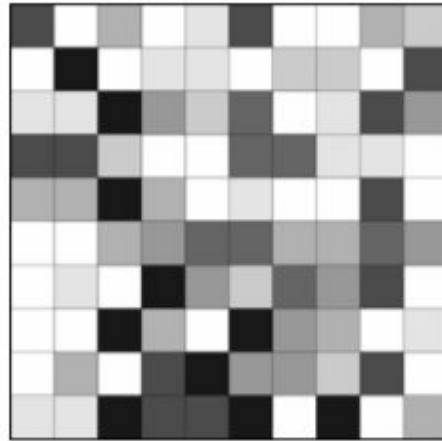




Types of Digital Image

- **BINARY IMAGE**– The binary image as its name suggests, contain only two pixel elements i.e 0 & 1, where 0 refers to black and 1 refers to white. This image is also known as Monochrome.
- **BLACK AND WHITE IMAGE**– The image which consist of only black and white color is called BLACK AND WHITE IMAGE.
- **8 bit COLOR FORMAT**– It is the most famous image format. It has 256 different shades of colors in it and commonly known as Grayscale Image. In this format, 0 stands for Black, and 255 stands for white, and 127 stands for gray.
- **16 bit COLOR FORMAT**– It is a color image format. It has 65,536 different colors in it. It is also known as High Color Format. In this format the distribution of color is not as same as Grayscale image.

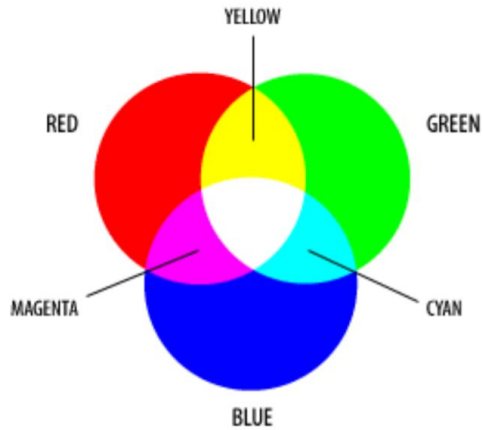
Black and White Image



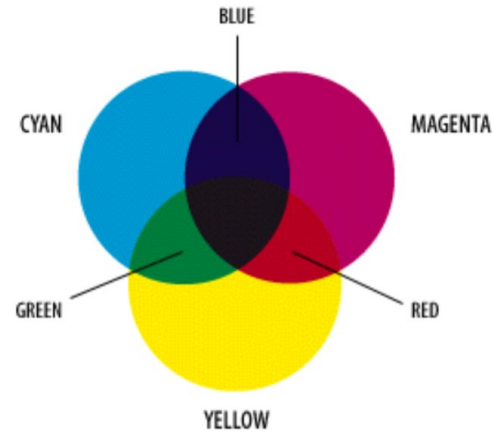
254	107
255	165

Figure 2: Each pixel has a value from 0 (black) to 255 (white). The possible range of the pixel values depend on the colour depth of the image, here 8 bit = 256 tones or greyscales.

Color Image Channel

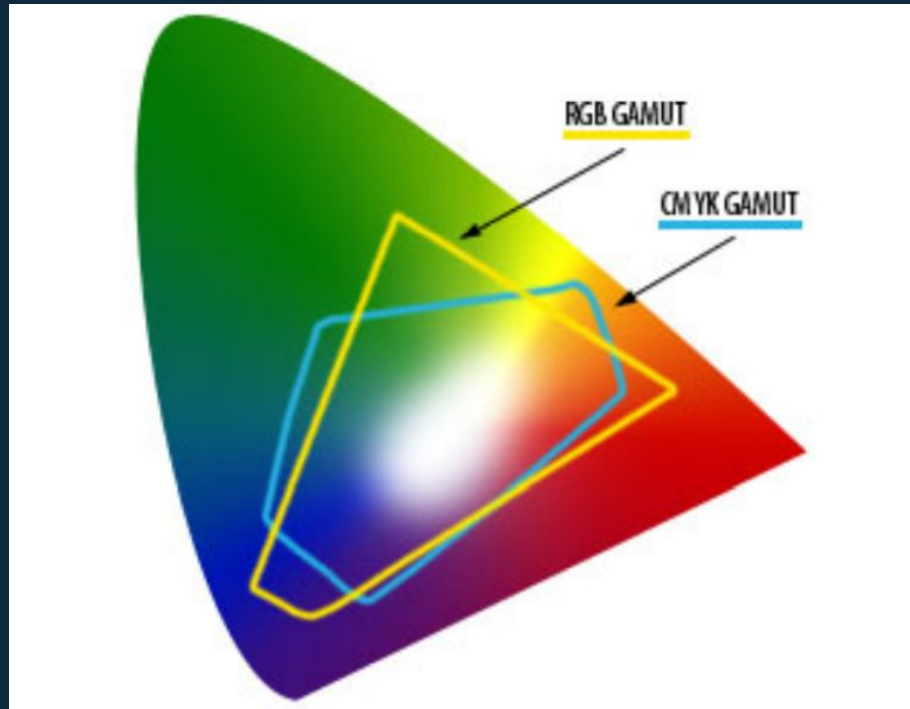


RGB Image



CMYK Image

CIE Chromaticity Diagram





OpenCV





What is OpenCV

OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision.

It Contains Many functions for image processing playing with images for making Image Manipulation easy.

It's an Open Source Library available for both c++ and Python.

We will be using python in these workshop.





Reading Image

Syntax: `cv2.imread(path, flag)`


Parameters:

path: A string representing the path of the image to be read.

flag: It specifies the way in which image should be read. It's default value is `cv2.IMREAD_COLOR`

- **cv2.IMREAD_COLOR:** It specifies to load a color image. Any transparency of image will be neglected. It is the default flag. Alternatively, we can pass integer value `1` for this flag.
- **cv2.IMREAD_GRAYSCALE:** It specifies to load an image in grayscale mode. Alternatively, we can pass integer value `0` for this flag.
- **cv2.IMREAD_UNCHANGED:** It specifies to load an image as such including alpha channel. Alternatively, we can pass integer value `-1` for this flag.

Return Value: This method returns an image that is loaded from the specified file. (numpy array)





namedWindow

Syntax: `cv2.namedWindow(name, flag)`

Parameters:

path: A string representing the name of window.

flag: It specifies the shape of Image.

- **WINDOW_NORMAL or WINDOW_AUTOSIZE:** WINDOW_NORMAL enables you to resize the window, whereas WINDOW_AUTOSIZE adjusts automatically the window size to fit the displayed image (see `imshow`), and you cannot change the window size manually.

Return Value: This method returns an window object which can be used to show any image.





waitKey

Syntax: `cv2.waitKey(value)`

Parameters:

value: *int* represented time to wait in milliseconds. If zero then the coe waits there as long as all image windows are opened.

Return Value: *Stops the code at that point for given amount of time*





Make a 200 pixel radius
circle in 800*800 pixel

- ◇ Create a numpy array of all 255 you can use `np.full`
- ◇ Move through every pixel check for condition
- ◇ Create a namedWindow, display image, set `waitKey`





Make a ChessBoard in
800*800 pixel image

◇ Same as previous one





createTrackbar

OpenCV provides function named as createTrackbar which can be used to create a trackbar. Trackbar is attached to a window .It is used to change the value of one of the variable mostly threshold so that we can find a perfect desirable range of color code.

Syntax:

```
cv2.createTrackbar(trackbar_name,title_window,default_value,max_value,callable_func)
```

Parameters:

title_window: name of the window that must contain that trackbar

max_value: max value of trackbar

callable_func: Callable function of the form `def call_func(val):` where val is a int number

Return Value: Creates a trackbar on specified window





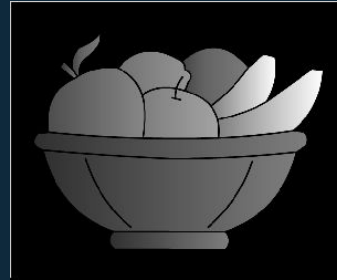
Conversion from RGB to Grayscale





Method 1

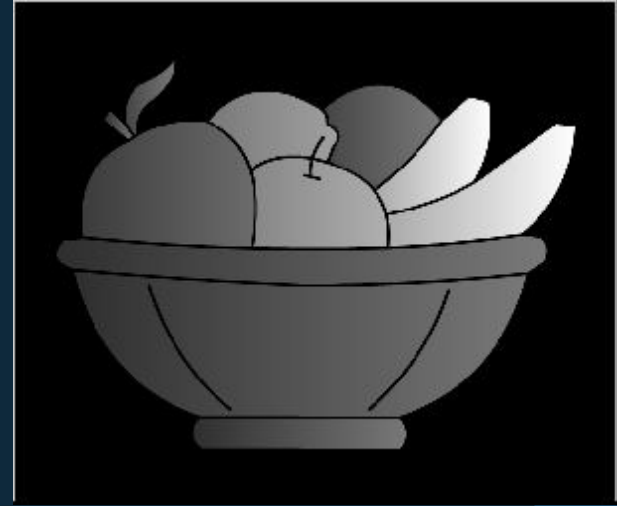
Take average of all three value.
That is new value = $(R+B+G)/2$



Method 2

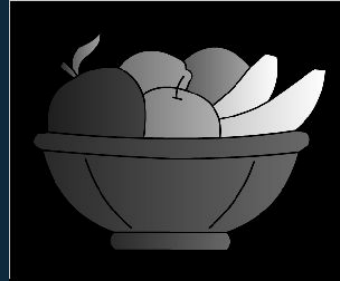
Since Human eye can see Green
Color more clearly and Blue color least

NEw value can be calculated as
 $\text{Grey} = 0.21R + 0.72G + 0.07B$
 $\text{Grey} = 0.299R + 0.587G + 0.114B$



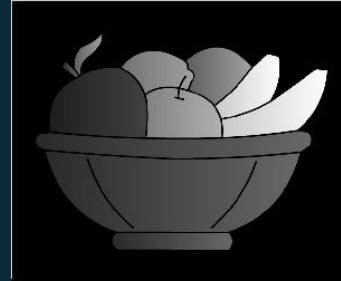
Method 3

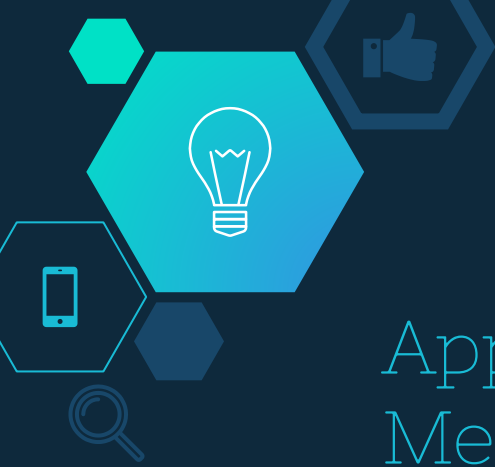
Take average of all three value.
Take mean of greatest and lowest value
$$\text{Grey} = (\max(r,b,g) + \min(r,b,g)) / 2$$



Method 3

Take average of all three value.
Take mean of greatest and lowest value
$$\text{Grey} = (\max(r,b,g) + \min(r,b,g)) / 2$$





Apply Method1, Method2, Method3 on a given Image

You can create 3 simple functions in a same file and display all 3 images at a time or create a 3 separate files

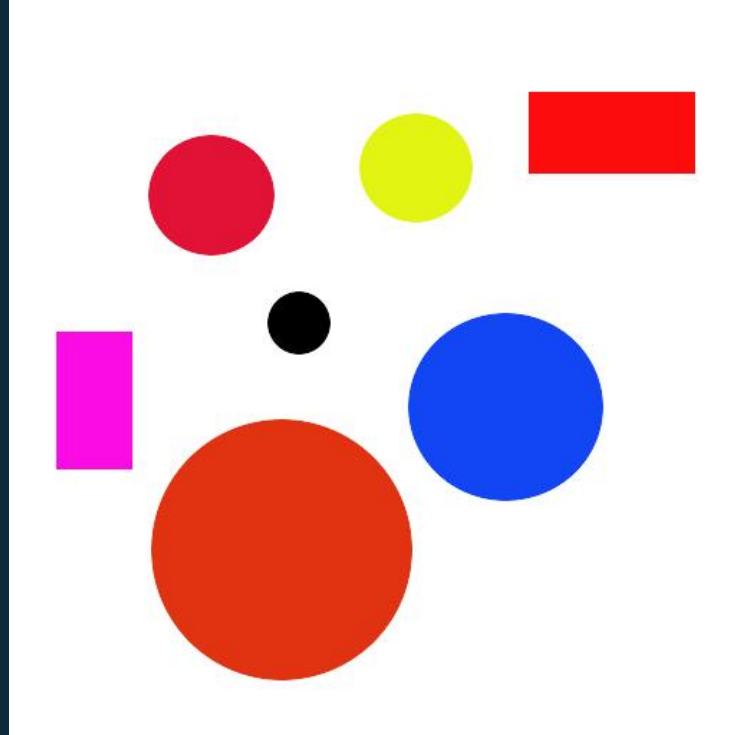




Colour Extraction

Extracting a range of colour/s from an image.

We will try and extract
each red object
individually





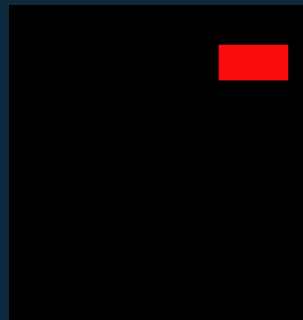
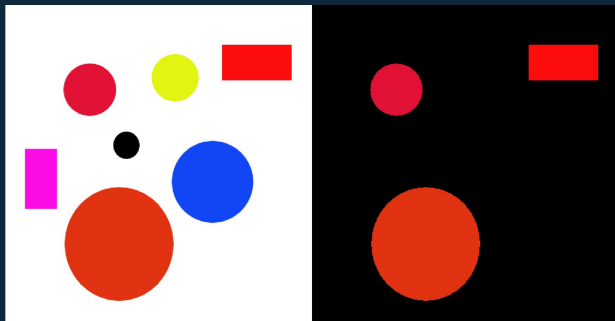
Some Suggestions

- ◇ Set Thresholds for max & min values per channel
- ◇ Use Trackbars to change them(Thresholds) and see the changes in real time.
- ◇ Display original image for reference.

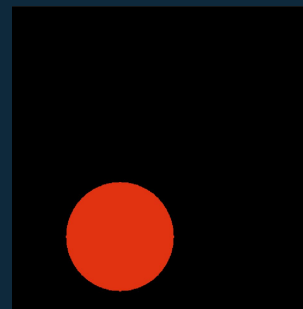


Expected Outputs

Original Image(Left) with All Red
objects detected(Right)



Individual Detected Objects

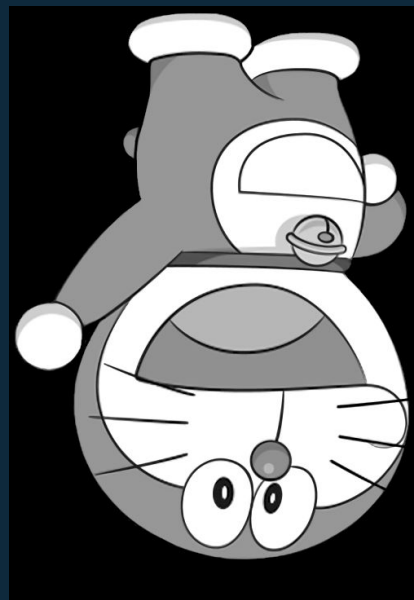




Geometric Operations



Flip Image





cv2.flip

Syntax: `cv2.flip(src, flipCode)`

Parameters:

src: *Input array.*

flipCode: 0 : Vertical 1:Horizontal -1:Both

Return Value: *It returns an image.*



Image Padding

Add borders to images of suitable colors.





cv2.copyMakeBorder

Syntax: *cv2.copyMakeBorder(src, top, bottom, left, right, borderType, value)*

Src: original image

Return Value: *It returns an image.*





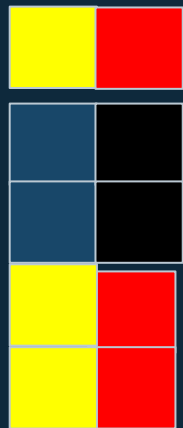
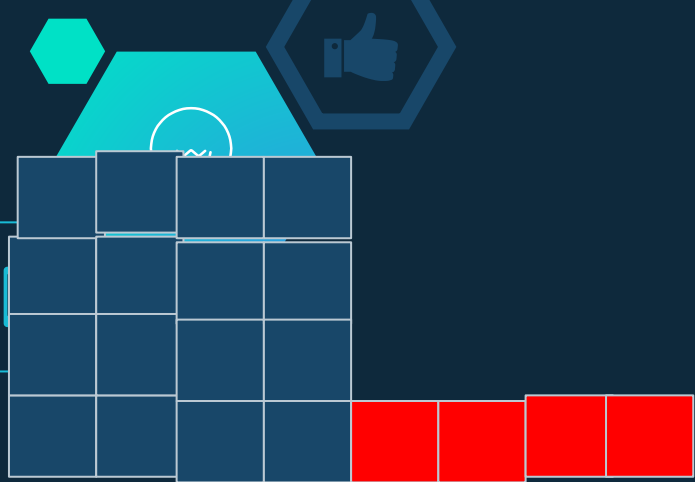
Image Resizing



Upsizing Images

2X2 --- 4X4 (2,2)

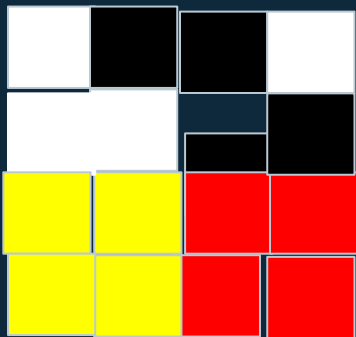
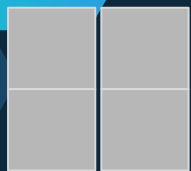
(N,M) N,M>=1



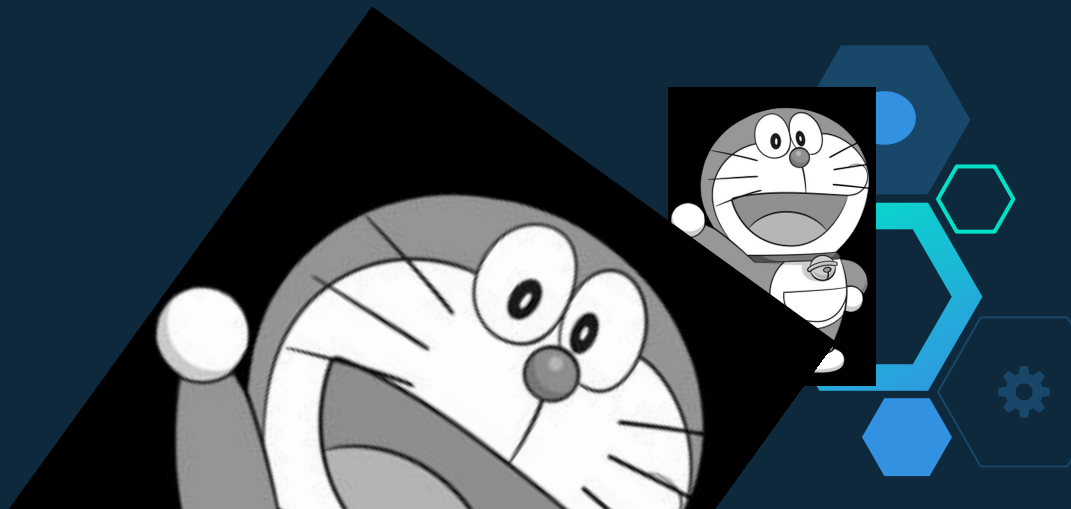
Downsizing Images

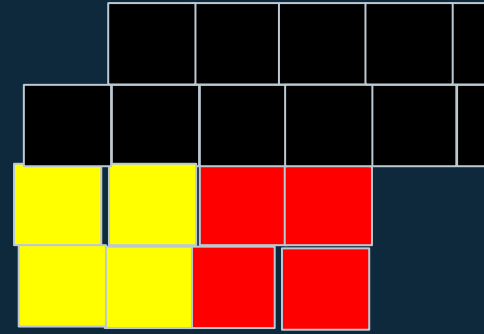
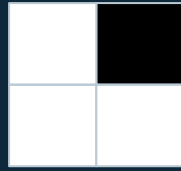
4X4 -----> 2X2

$(1/n, 1/m)$ $n, m \geq 1$



36





37

Resizing Images

Q. How to resize an Image of size (4,4) - \rightarrow (3,3)

4,4 \rightarrow 1x1 \rightarrow (3,3)

4,4 \rightarrow 12,12 \rightarrow (3,3)

4,4 \rightarrow





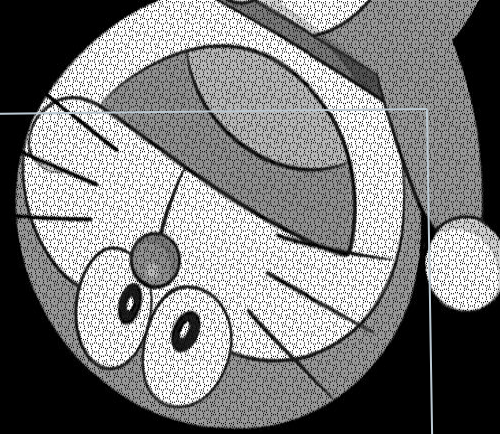
Resizing Images

Q. How to resize an Image of size $(4,4) \rightarrow (3,3)$

Ans. $(4,4) \rightarrow (12,12) \rightarrow (3,3)$



Rotation



1. Find the edge length(L)
2. Make a new image of LxL
3. Find the centre of original image
4. For all pairs of x,y in original image:
 - Compute distance from centre
 - Compute angle using \tan^{-1}
 - Find new rotated coordinates
 - Shift coordinates to have centre of new image

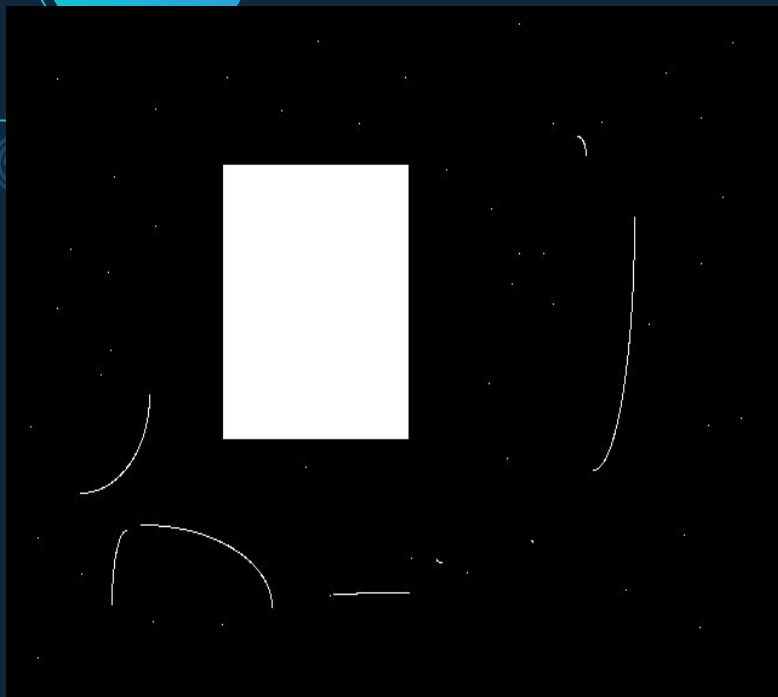




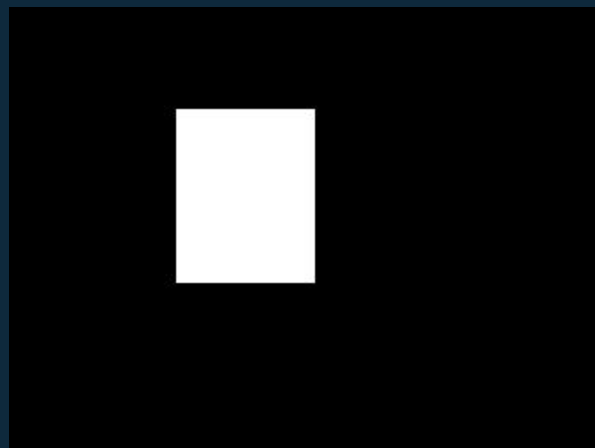
Morphological Operations



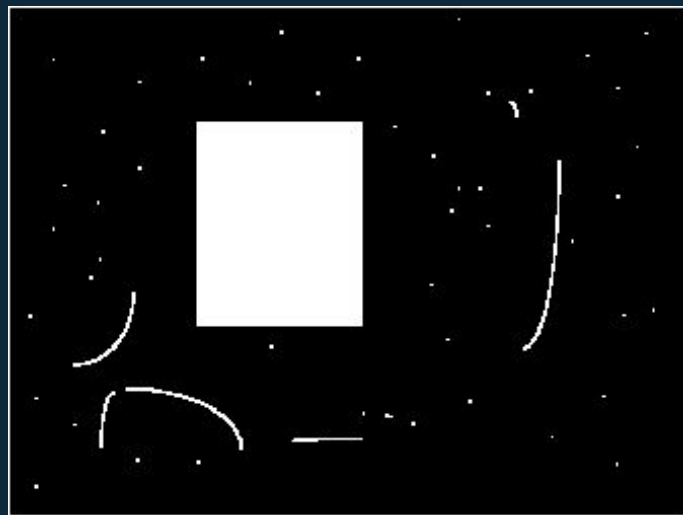
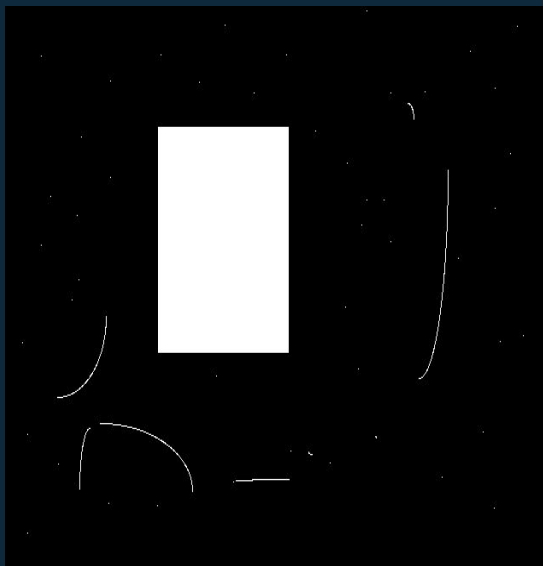
Erosion



For any pixel in original image,
Check if it has any black pixel around it,
If yes:
 Make it black
else :
 continue



Dilation





Histograms

It is a graph showing the number of pixels in an image at each different intensity value found in that image.



A decorative graphic on the left side of the slide consists of a large cyan hexagon in the center. Surrounding it are several smaller hexagons in various shades of blue and cyan. Some of these hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, a gear, and a speech bubble. There is also a small network-like icon with a central node and three connecting lines.

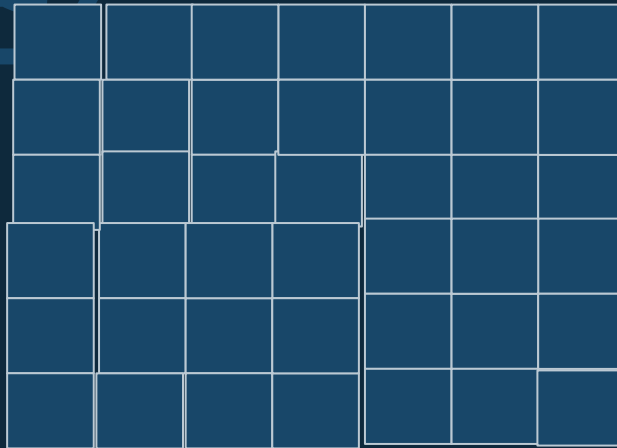
Spatial Features

https://opencv24-python-tutorials.readthedocs.io/en/stable/py_tutorials/py_imgproc/py_filtering/py_filtering.html

Convolutions: <https://setosa.io/ev/image-kernels/>



Blur



1/273	4/273	7/273	4/273	1/273
4/273	16/273	26/273	16/273	4/273
7/273	26/273	41/273	26/273	7/273
4/273	16/273	26/273	16/273	4/273
1/273	4/273	7/273	4/273	1/273

-1/9
-1/9
-1/9

- *Average Blur:*

Also known as Low pass filtering and

Smoothing filter

Using Average Kernels

- *Gaussian Blur:*

Using Gaussian Kernels

Used for noise reduction and smoothing





Gaussian Function

$$G_{2D}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

How to obtain Gaussian Kernel Using Gaussian Function:

https://www.cs.auckland.ac.nz/courses/compsci373s1c/PatricesLectures/Gaussian%20Filtering_1up.pdf





Median Filtering

- ◇ Used to remove Salt & Pepper Noise in Grayscale images.
- ◇ Median of all values covered by kernel is kernel's output.





Try all Filters





Edge Detection

<https://cse442-17f.github.io/Sobel-Laplacian-and-Canny-Edge-Detection-Algorithms/>





More on Canny

https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html





Try Sobel Filter

