**Open CV Table**

1. What is Computer Vision
2. What is OpenCV
3. Read and Show Image
4. Show Multiple Images and Slide Show
   1. Show Multiple Images
   2. Slide Show
5. Text Over an Image
6. Draw Line and Rectangle on Image
   1. Draw Line
   2. Rectangle on Image
7. Draw Circle and Ellipse on Image
   1. Draw Circle on Image
   2. Draw Ellipse on Image
8. Draw Polygons on Image
9. Arithmetic Operations on images using opencv
10. Bitwise operation on images using opencv
11. Edge Detection in an image using opencv
12. Image Scaling and Rotating using opencv
    1. Image Scaling
    2. Image Rotating
13. Image Blurring using opencv
14. IMWRITE Method using opencv
15. Make Border using opencv
16. Play a Video using opencv
17. Capture video from camera using opencv
18. Slow and Fast Motion Video
19. Morphological Operations using opencv
20. Image Pyramid using opencv
21. Image Translation using opencv
22. Geometric Transformations of Images
23. Background Subtraction using opencv
24. Extract Images from Video using opencv
25. cvtColor method opencv
26. Crop Image using opencv
27. Create Blank Image opencv
28. Color Picker using opencv
29. getTrackbarPos() function in opencv
30. Region of Interest using opencv
31. Flip, Rotate & Transpose using opencv
32. Saving a video using opencv
33. Filter color with opencv (live object filter)
34. Perspective Transformation opencv
35. Threshold OpenCV
    1. Simple Threshold
    2. Otsu’s Binarization Threshold
36. Analyze an Image using Histogram
37. Histograms Equalization in opencv
38. Mouse Event bindings suing opencv
39. Image Contours using opencv
40. Contour Moments and Convex Hull using opencv
41. Object Detections using Contours by Webcam imp
42. Image Background Removal Histogram
43. Hough Transformation Lines
44. Template Matching using opencv
45. Hough Circle Detection
46. Grab cut Algorithm for background Change
47. Video Background Removal using Algorithm
48. Object Tracking and Detection
49. Corner Detecting (Harris Corner Detection)

**Computer Vision**

# What is Computer Vision

Computer Vision is the field of computer science that focuses of creating digital systems that can process, analyze, and make sense of visual data (images or videos) in the same way that humans do.

## Computer Vision Systems Task

* Object Classification
* Object Identification
* Object Tracking
* Image Restoring
* Feature Matching
* Video motion analysis

## How does Computer Vision Work?

**Acquiring an image**

Images, even large sets, can be acquired in real-time through video, photo, or 3D technology for analysis

**Processing the image**

Deep learning models automate much of this process, but the models are often trained by first being fed a thousand of labeled or pre-identified images.

**Understanding the image**

The final step is the interpretative step, where an object is identified or classified.

**OpenCV**

# What is OpenCV

* OpenCV is python open-source library, which is used for computer vision in artificial Intelligence, Machine Learning, face recognition, etc.
* It is includes several hundreds of computer vision algorithms.
* I t has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS.
* OpenCV is written natively in C++.
* Initial Release in June 2000: 20 years ago

## How OpenCV works

There are two common ways to identify the images:

* Grayscale
* RGB

## Why OpenCV is used for Computer Vision

* OpenCV is available for free of cost.
* Since the OpenCV library is written in C/C++, so it is quit fast. Now it can be used with python.
* It Requires less RAM to usage, it maybe 60-70 MB.
* Computer Vision is portable as OpenCV and can run on any device that can run on C.

## Prerequisite

Python Numpy

SciPy Matrix Matplotlib

## OpenCV Modules

OpenCV

core imgproc imgcodecs highgui video

calib3d features2d objdetect dnn ml

flann photo stitching shape superres

Videostab viz

# Read and Show Image

Cv2.IMREAD\_COLOR: It specifies to load a color image. Any transparency of image will be neglected. It is a default flag. Alternatively, we can pass integer value 1 for this flag.

Cv2.IMREAD\_GRAYSCALE: It specifies to load an image in grayscale color. Alternatively, we can pass integer value 1 for this flag.

Cv2.IMREAD\_UNCHANGED: It specifies to load an image as show including alpha channel. Alternatively, we can pass integer value -1 for this flag.

## Practical – read and show image

import cv2

img = cv2.imread("img\_1.jpg") # image read

cv2.imshow("maheen",img) # image show

cv2.waitKey(0) # image frame wait Time - ms and 0 pass keyboard any key pass

cv2.destroyAllWindows() # all windows close

# cv2.destroyWindow() # only one window close

**Multiple Images**

img = cv2.imread("img\_1.jpg") # image read

cv2.imshow("maheen haroon", img)

cv2.imshow("maheen",img) # image show

cv2.waitKey(0) # image frame wait Time - ms and 0 pass keyboard any key pass

cv2.destroyAllWindows() # all windows close

# cv2.destroyWindow("maheen") # only one window close

# Show multiple images and Slide Show

import cv2

import numpy as np

import os

img = cv2.imread(r"C:\Users\Z\Desktop\AI Courses\Computer Vision\images\img\_2.jpg")

re\_img = cv2.resize(img, (300, 600)) # image resize

cv2.imshow('maheen', re\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

## Show Multiple Images

img = cv2.imread(r"C:\Users\Z\Desktop\AI Courses\Computer Vision\images/img\_2.jpg")

re\_img = cv2.resize(img, (150,250))

h = np.hstack((re\_img, re\_img, re\_img)) # set horizontal images

v = np.vstack((h,h)) # set vertical images

cv2.imshow('maheen', v)

cv2.waitKey(0)

cv2.destroyAllWindows()

## Slide Show

list\_name = os.listdir(r"C:\Users\Z\Desktop\AI Courses\Computer Vision\images")

for name in list\_name:

path = "C:\\Users\\Z\\Desktop\\AI Courses\\Computer Vision\\images"

img\_name = path + '\\' + name

img = cv2.imread(img\_name)

img = cv2.resize(img, (300,500))

cv2.imshow('Maheen Haroon', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Text over an Image

img\_get = cv2.imread(r"C:\Users\Z\Desktop\AI Courses\Computer Vision\images/img\_2.jpg")

img\_get = cv2.resize(img\_get, (600, 1000))

txt = cv2.putText(img = img\_get,

text = "Maheen Haroon", # Text

org = (50,50), # Text Margin

fontFace = cv2.FONT\_HERSHEY\_DUPLEX, # Text Font\_family

fontScale = 2, # Text Size

color = (0,0,255), # Text Color

thickness = 1, # Text Style

lineType = cv2.LINE\_AA,

bottomLeftOrigin = False) # or True

cv2.imshow('maheen', img\_get)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Draw Line and Rectangle on Image

## Draw Line

old\_img = cv2.imread(r"C:\Users\Z\Desktop\AI Courses\Computer Vision\images/img\_1.jpg")

old\_img = cv2.resize(old\_img, (300,400))

# draw line

new\_img = cv2.line(img = old\_img, pt1= (120,40), pt2= (190,40), color = (0,255,0), thickness=4, lineType = 4)

cv2.imshow('Maheen', new\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

## Rectangle on Image

old\_img = cv2.imread(r"C:\Users\Z\Desktop\AI Courses\Computer Vision\images/img\_1.jpg")

old\_img = cv2.resize(old\_img, (300, 400))

# The org parameter should be a tuple

txt\_img = cv2.putText(img=old\_img, text='Maheen', org=(115, 18), fontFace=1,

fontScale=1, color=(0, 255, 0), thickness=1, lineType=cv2.LINE\_AA)

# Rectangle on Image

new\_img = cv2.rectangle(img=txt\_img, pt1=(115, 20), pt2=(205, 130), color=(0, 255, 0), thickness=1, lineType=cv2.LINE\_AA)

cv2.imshow('Maheen', new\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Draw Circle and Ellipse on Image

## Draw Circle on Image

old\_img = cv2.imread(r"C:\Users\Z\Desktop\AI Courses\Computer Vision\images/img\_1.jpg")

old\_img = cv2.resize(old\_img, (300,400))

new\_img = cv2.circle(img=old\_img, center=(160,70), radius=55, color=(0, 255, 0), thickness=2, lineType=cv2.LINE\_AA)

cv2.imshow("Maheen", new\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

## Draw Ellipse on Image

old\_img = cv2.imread(r"C:\Users\Z\Desktop\AI Courses\Computer Vision\images/img\_2.jpg")

old\_img = cv2.resize(old\_img, (300,500))

txt\_img = cv2.putText(img = old\_img, text="Maheen", org=(200,145), fontFace=1, fontScale=1, color=(0,255,0), thickness=2, lineType=4)

# Draw Ellipse on Image

new\_img = cv2.ellipse(img = txt\_img, center=(160,120), axes=(55,70),angle = 30, startAngle = 0, endAngle = 345,

color=(0, 255, 0), thickness=2, lineType=cv2.LINE\_AA)

cv2.imshow("Maheen", new\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Draw Polygons on Image

old\_img = cv2.imread(r"C:\Users\Z\Desktop\AI Courses\Computer Vision\images/img\_6.jpg")

old\_img = cv2.resize(old\_img, (400,300))

new\_img = cv2.polylines(img=old\_img,

pts=[np.array([[145,40],[135,80],[160,125],[190,125],[200,100], [200,40]])],

isClosed=True, color=(0,255,0), thickness=2, lineType=2)

cv2.imshow("Woman",new\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Arithmetic Operations on Images using opencv

img1 = cv2.imread(r"C:\Users\Z\Desktop\AI Courses\Computer Vision\images/img\_2.jpg")

img2 = cv2.imread(r"C:\Users\Z\Desktop\AI Courses\Computer Vision\images/img\_9.jpg")

img1 = cv2.resize(img1, (300,500))

img2 = cv2.resize(img2, (300,500))

#Merge Images

new\_img = cv2.addWeighted(img1, 1, img2, 1, 1)

# Subract Images

new\_img = cv2.subtract(img2,img1)

cv2.imshow("Maheen Haroon", new\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Bitwise Operation on Images using opencv

img1 = cv2.imread("images\img\_10.png")

img2 = cv2.imread("images\img\_11.png")

img1 = cv2.resize(img1, (300,500))

img2 = cv2.resize(img2, (300,500))

# And Operation

new = cv2.bitwise\_and(img1, img2)

# Or Operation

new = cv2.bitwise\_or(img1, img2)

# Xor Operation

new = cv2.bitwise\_xor(img1, img2)

# Not Operation

new = cv2.bitwise\_not(img1)

h = np.hstack((img1,img2,new))

cv2.imshow("Bitwise Image", h)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Edge Detection in an image using opencv

img = cv2.imread("images\img\_1.jpg")

img = cv2.resize(img, (300,400))

print(img.shape)

new\_img = cv2.Canny(img, 100, 200, apertureSize=7, L2gradient=True)

print(new\_img.shape)

cv2.imshow("Maheen", img)

cv2.imshow("Maheen1", new\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Image Scaling and Rotating using opencv

## Image Scaling

img = cv2.imread("images\img\_4.jpg")

re\_img = cv2.resize(img, (550,400))

cv2.imshow("Random", re\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

## Image Rotating

img = cv2.imread("images\img\_4.jpg")

re\_img = cv2.resize(img, (500,500))

(w,h) = re\_img.shape[0], re\_img.shape[1]

center = (w/2, h/2)

matrix = cv2.getRotationMatrix2D(center, 45, 1)

rotated = cv2.warpAffine(re\_img, matrix, (h,w))

cv2.imshow("Random", rotated)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Image Blurring using opencv

## Gaussian Blurring

**Gaussian Blurring:** Gaussian blur is the result of blurring an image by a Gaussian function.

It is widely used effect in graphics software, typically to reduce image noise and reduce detail.

It is also used as a preprocessing stage before applying our machine learning or deep learning models.

## Median Blur

**Median Blur:** The Median filter is a non-linear digital filtering technique, often use to remove noise from an image or signal.

Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise.

It is one of the best algorithms to remove salt and pepper noise.

## Bilateral Blur

**Bilateral Blur:** A bilateral filter is a non-linear, edge-preserving, and noise-reducing smoothing filter for images.

It replace the intensity of each pixel with a weighted average of intensity values from nearby pixels.

This weight can be based on a Gaussian distribution.

Thus, sharp edges are preserved while discarding the weak ones.

## Practical – image blurring

org\_img = cv2.imread("images\img\_6.jpg")

re\_img = cv2.resize(org\_img, (300,200))

# Gaussian Blur

g = cv2.GaussianBlur(re\_img, (7,7), 0)

# Median Blur

m = cv2.medianBlur(re\_img, 3) # 3 or 5

# Bilateral Blur

b = cv2.bilateralFilter(re\_img,5, 100,100)

h = np.hstack((re\_img,g,m,b))

cv2.imshow("Random", h)

cv2.waitKey(0)

cv2.destroyAllWindows()

# IMWRITE Method using opencv

img1 = cv2.imread("images\img\_9.jpg")

img2 = cv2.imread("images\img\_1.jpg")

img3 = cv2.imread("images\img\_2.jpg")

img4 = cv2.imread("images\img\_7.jpg")

re\_img1 = cv2.resize(img1, (300,300,))

re\_img2 = cv2.resize(img2, (300,300,))

re\_img3 = cv2.resize(img3, (300,300,))

re\_img4 = cv2.resize(img4, (300,300,))

h = np.hstack((re\_img1, re\_img2))

h1 = np.hstack((re\_img3,re\_img4))

v = np.vstack((h,h1))

cv2.imwrite("images\merge.jpg", v)

cv2.imshow("merge",v)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Make Border using opencv

img = cv2.imread("images\img\_7.jpg")

img = cv2.resize(img, (400,300))

# cv2.BORDER\_CONSTANT, cv2.BORDER\_REFLECT, cv2.BORDER\_REFLECT\_101, cv2.BORDER\_DEFAULT, cv2.BORDER\_REPLICATE, cv2.BORDER\_ISOLATED

new\_img = cv2.copyMakeBorder(img, 20,20,20,20, cv2.BORDER\_REPLICATE)

cv2.imshow("Random", new\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Play a Video using opencv

cap = cv2.VideoCapture(r"images\video\_1.mp4") # Use raw string or forward slashes

while cap.isOpened():

ret, frame = cap.read()

if ret:

frame = cv2.resize(frame, (600, 500))

cv2.imshow("Cricket", frame)

# Wait for 25ms and check if 'p' is pressed to pause/exit

if cv2.waitKey(25) & 0xFF == ord('p'):

break

else:

break

cap.release()

cv2.destroyAllWindows()

# Capture video from camera using opencv

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

if ret == True:

frame = cv2.resize(frame, (500, 500))

cv2.imshow("Camera", frame)

if cv2.waitKey(25) & 0xFF == ord("p"):

break

else:

break

cap.release()

cv2.destroyAllWindows()

# Slow and Fast Motion Video

cap = cv2.VideoCapture(r"images\video\_1.mp4")

while cap.isOpened():

ref, frame = cap.read()

if ref == True:

frame = cv2.resize(frame, (500,500))

cv2.imshow("Cricket", frame)

# if wait key is less than 25 video runing fast and if wait key is greater than 25 video runing slow.

if cv2.waitKey(25) & 0xff == ord("p"):

break

else:

break

cap.release()

cv2.destroyAllWindows()

# Morphological Operations using opencv

## Morphological Operations

Morphological Operations are used to extract images components that are useful in the representation and description of region shape.

It is typically performed on binary images.

The Morphological Operations we will covering includes:

* Erosion
* Dilation
* Opening
* Closing
* Morphological gradient
* Black hat
* Top hat (also called “White hat”)

## Practical

img = cv2.imread(r"images\img\_13.png")

img1 = cv2.imread("images\img\_14.jpeg")

img = cv2.resize(img, (500,500))

img1 = cv2.resize(img1, (500,500))

m = np.ones((40,40),np.int8)

# Erosion Image

er = cv2.erode(img,m, iterations=1)

# Dilation Image

di = cv2.dilate(img,m, iterations=1)

# Opeing Image

op = cv2.morphologyEx(img1, cv2.MORPH\_OPEN, m,iterations=1)

# Closing Image

cl = cv2.morphologyEx(img1, cv2.MORPH\_CLOSE, m, iterations=1)

# Morphological Gradient

gr = cv2.morphologyEx(img, cv2.MORPH\_GRADIENT, m, iterations=1)

# Top / White Hat

tp = cv2.morphologyEx(img, cv2.MORPH\_TOPHAT, m, iterations=1)

# Black Hat

bl = cv2.morphologyEx(img, cv2.MORPH\_BLACKHAT, m, iterations=1)

cv2.imshow("img", img)

cv2.imshow("Erosion",er)

cv2.imshow("Dilation", di)

cv2.imshow("img1", img1)

cv2.imshow("Opening", op)

cv2.imshow("Closing", cl)

cv2.imshow("Morphological Gradient", gr)

cv2.imshow("Top / White Hat", tp)

cv2.imshow("Black Hat", bl)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Image Pyramid using opencv

Work with images with default resolution but many times we need to change the resolution (lower it) or resize the original image in that case image pyramids come handy.

img = cv2.imread("images\img\_5.JPG")

# Down Pyramid

new = cv2.pyrDown(img)

new1 = cv2.pyrDown(new)

# Up Pyramid

up\_img = cv2.pyrUp(img)

cv2.imshow("img", img)

cv2.imshow("Down Pyramid", new)

cv2.imshow("Down Pyramid1", new1)

cv2.imshow("Up Pyramid", up\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Image Translation using opencv

img = cv2.imread("images\img\_7.jpg")

img = cv2.resize(img, (500,350))

m = np.float32([[1,0,100],[0,1,50]])

new\_img = cv2.warpAffine(img,m,(500,500))

cv2.imshow("img", img)

cv2.imshow("m", m)

cv2.imshow("new\_img", new\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Geometric Transformations of Images

## Transformations

* Scaling
* Translation
* Rotation

Affine Transformation: All parallel lines in the original image will still be parallel in the output image.

To find the transformation matrix, we need three points from the input image and their corresponding locations in the output image.

Then cv2.getAffine Transform will create a 2x3 matrix which is to be passed to cv2.warpAffine.

Perspective Transformation: you need a 3x3 transformation matrix. Straight lines will remain straight even after the transformation.

You need 4 points on the input image and corresponding points on the output image.

Among these 4 points, 3 of them should not be collinear.

Then the transformation matrix can be found by the function cv2.getPerspectiveTransform.

Then apply cv2.warpPerspective with this 3x3 transformation matrix.

# Background Subtraction using opencv

org\_v = cv2.VideoCapture(r"images\video\_1.MP4")

sub\_m = cv2.createBackgroundSubtractorMOG2()

while True:

ret, frame = org\_v.read()

if ret == True:

frame = cv2.resize(frame, (700,500))

sub\_v = sub\_m.apply(frame)

cv2.imshow("Sub Video", sub\_v)

cv2.imshow("Video Play", frame)

if cv2.waitKey(25) & 0xff == ord('p'):

break

else:

break

org\_v.release()

cv2.destroyAllWindows()

# Extract Images from Video using opencv

cap = cv2.VideoCapture(r"images\video\_1.mp4")

c =0

while True:

r, frame = cap.read()

if r == True:

cv2.imshow("Video", frame)

filename = "New folder//image\_" + str(c) + ".png"

cv2.imwrite(filename, frame)

c = c + 1

if cv2.waitKey(25) & 0xff == ord('p'):

break

else:

break

cap.release()

cv2.destroyAllWindows()

# CvtColor Opencv

org = cv2.imread(r"images/img\_4.jpg")

org = cv2.resize(org, (500, 350))

new = cv2.cvtColor(org, cv2.COLOR\_BGR2HLS)

cv2.imshow("Original Image", org)

cv2.imshow("New Image", new)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Crop Image using opencv

img = cv2.imread("images\img\_5.jpg")

img = cv2.resize(img, (500,350))

# [y1:y2, x1:x2]

crop = img[5:150,175:320]

cv2.imshow("Image", img)

cv2.imshow("Crop", crop)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Create Blank Image opencv [white and black image]

white\_img = np.ones((500,500,3), np.uint8)\*255

cv2.imshow("White Image", white\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

black\_img = np.zeros((500,500,3), np.uint8)\*255

cv2.imshow("Black Image", black\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Color Picker using opencv

def image(x):

pass

img = np.ones((500,500,3), np.uint8)\*255

cv2.namedWindow("colour")

cv2.createTrackbar("R", "colour",0,255, image)

cv2.createTrackbar("G", "colour",0,255, image)

cv2.createTrackbar("B", "colour",0,255, image)

while True:

cv2.imshow("colour", img)

if cv2.waitKey(1) & 0xff == ord('p'):

break

r = cv2.getTrackbarPos("R", "colour")

g = cv2.getTrackbarPos("G", "colour")

b = cv2.getTrackbarPos("B", "colour")

img[:] = [b, g, r]

cv2.destroyAllWindows()

# getTrackbarPos() function in opencv

def deff(x):

pass

img = np.zeros((500,500,3), np.uint8)\*255

cv2.namedWindow("bar")

cv2.createTrackbar("on", "bar", 0,200, deff)

while True:

cv2.imshow("bar", img)

if cv2.waitKey(1) & 0xff == ord("p"):

break

on = cv2.getTrackbarPos("on", "bar")

img[:] = [0,100,on]

cv2.destroyAllWindows()

# Region of Interest using opencv

img = cv2.imread("images\img\_5.jpg")

# [y1:y2, x1:x2]

crop = img[5:145,215:315]

img[5:145,315:415] = crop # right

img[5:145,415:515] = crop # right

img[145:285,415:515] = crop # right down

img[5:145,115:215] = crop # left

img[5:145,15:115] = crop # left

img[145:285,15:115] = crop # left down

cv2.imshow("img", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Flip, Rotate & Transpose using opencv

img = cv2.imread("images\img\_6.jpg")

# Flip

img = cv2.flip(img, 1) # y axis

img = cv2.flip(img, 0) # x axis

img = cv2.flip(img, -1) # x, y axis

# Rotate

img = cv2.rotate(img, cv2.ROTATE\_90\_COUNTERCLOCKWISE)

# Transpose

img = cv2.transpose(img)

cv2.imshow("Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Saving a video using opencv

cap = cv2.VideoCapture(0)

f = cv2.VideoWriter\_fourcc(\*'XVID')

out = cv2.VideoWriter("demo.mp4",f,25.0,(640,480)) # add 0 for grayscale video

while cap.isOpened():

ret, frame = cap.read()

if ret == True:

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) # add this for gray scale video

frame = cv2.flip(frame,1)

cv2.imshow("Video", frame)

out.write(frame)

if cv2.waitKey(1) & 0xff == ord('p'):

break

else:

break

cap.release()

out.release()

cv2.destroyAllWindows()

# Filter color with opencv (live object filter)

cap = cv2.VideoCapture(0)

def demo(x):

pass

cv2.namedWindow("demo")

cv2.createTrackbar("lb","demo", 0,255, demo)

cv2.createTrackbar("lg","demo", 0,255, demo)

cv2.createTrackbar("lr","demo", 0,255, demo)

cv2.createTrackbar("ub","demo", 255,255, demo)

cv2.createTrackbar("ug","demo", 255,255, demo)

cv2.createTrackbar("ur","demo", 255,255, demo)

while cap.isOpened():

r, frame = cap.read()

if r == True:

img = cv2.resize(frame, (400,300))

hsv\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

Lb = cv2.getTrackbarPos("lb", "demo")

Lg = cv2.getTrackbarPos("lg", "demo")

Lr = cv2.getTrackbarPos("lr", "demo")

Ub = cv2.getTrackbarPos("ub", "demo")

Ug = cv2.getTrackbarPos("ug", "demo")

Ur = cv2.getTrackbarPos("ur", "demo")

lo = np.array([Lb, Lg, Lr])

up = np.array([Ub, Ug, Ur])

masks = cv2.inRange(hsv\_img, lo, up)

res = cv2.bitwise\_and(img, img, mask = masks)

cv2.imshow("res",res)

cv2.imshow("mask", masks)

cv2.imshow("hsv", hsv\_img)

cv2.imshow("Original", img)

if cv2.waitKey(1) & 0xff == ord('p'):

break

else:

break

cap.release()

cv2.destroyAllWindows()

# Perspective Transformation opencv

img = cv2.imread("images\img\_15.jpeg")

w,h = (500,600)

src1 = np.float32([[55, 14], [222, 76], [3, 173], [179, 223]])

dst1 = np.float32([[0, 0], [w, 0], [0, h], [w, h]]) # first point [0,0], sec [w,0], 3rd [0,h], 4th [w,h]

m = cv2.getPerspectiveTransform(src1, dst1)

new\_img = cv2.warpPerspective(img, m, (w, h))

cv2.imshow("image", img)

cv2.imshow("new\_image", new\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Threshold OpenCV

Threshold is one of the most common (and basic) segmentation techniques in computer vision and it allows us to separate the foreground (i.e., the objects that we are interested in) from the background of the image.

Threshold is the binarization of an image.

Threshold comes in three forms:

* Simple Threshold
* Otsu’s Threshold
* Adaptive Threshold

## Simple Threshold

img = cv2.imread("images\img\_7.jpg")

img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

\_, th = cv2.threshold(img, 100, 255, cv2.THRESH\_TOZERO)

cv2.imshow("Threshold Image", th)

cv2.imshow("Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

## Otsu’s Binarization Threshold

img = cv2.imread("images\img\_5.jpg")

img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# hist = cv2.calcHist([img], [0], None, [255], [0, 255])

# plt.plot(hist)

# plt.show()

\_, thr = cv2.threshold(img, 0,255, cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

cv2.imshow("Threshold", thr)

cv2.imshow("Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Analyze an Image using Histogram

import matplotlib.pyplot as plt

img = cv2.imread("images\img\_7.jpg")

# GrayScale Image

img1 = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

hist = cv2.calcHist([img1], [0], None, [255], [0,255])

plt.plot(hist)

plt.show()

# Color Image

b,r,g = cv2.split(img)

plt.hist(b.ravel(), 255, (0,255), color="blue")

plt.hist(g.ravel(), 255, (0,255), color="green")

plt.hist(r.ravel(), 255, (0,255), color="red")

plt.show()

cv2.imshow("Original Image", img)

# cv2.imshow("GrayScale Image", img1)

cv2.imshow("Blue Image", b)

cv2.imshow("Green Image", g)

cv2.imshow("Red Image", r)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Histograms Equalization in opencv

img = cv2.imread("images\img\_3.jpeg")

img = cv2.resize(img, (600,400))

gry = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Adaptive Equalization

hist = cv2.calcHist([gry], [0], None, [255], [0,255])

plt.plot(hist)

eq = cv2.equalizeHist(gry)

hist1 = cv2.calcHist([eq], [0], None, [255], [0,255])

plt.plot(hist1, c='red')

# CLAHE (contrast limited adaptive histrogram equalization)

cl = cv2.createCLAHE(clipLimit = 1, tileGridSize = (8,8))

f = cl.apply(gry)

plt.plot(f, c='green')

plt.show()

cv2.imshow("Original Image", img)

cv2.imshow("Adaptive Equalization", eq)

cv2.imshow("Clahe Image", f)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Mouse Event Bindings suing opencv

def python(event, x, y, flags, param):

if event == cv2.EVENT\_LBUTTONDOWN:

cv2.circle(img, (x, y), 5, (0, 0, 255), -1) # Filled circle

elif event == cv2.EVENT\_RBUTTONDOWN:

cv2.rectangle(img, (x, y), (x + 20, y + 20), (255, 0, 0), -1) # Filled rectangle

cv2.namedWindow("Moon")

img = np.ones((600, 600, 3), np.uint8) \* 255

cv2.setMouseCallback("Moon", python)

while True:

cv2.imshow("Moon", img)

key = cv2.waitKey(1) & 0xFF # Use waitKey(1) for continuous update

if key == ord('p'):

break

cv2.destroyAllWindows()

# Image Contours using opencv

img = cv2.imread("images\img\_16.png")

gry = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

r,thr = cv2.threshold(gry, 250,255, cv2.THRESH\_BINARY)

c, h = cv2.findContours(thr, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

new\_img = cv2.drawContours(img, c, -1, (255,0,0),2)

cv2.imshow("New Image", new\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Contour Moments and Convex Hull using opencv

img = cv2.imread("images\img\_18.jpg")

img = cv2.resize(img, (500,600))

gry = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

r, thr = cv2.threshold(gry, 200,255, cv2.THRESH\_BINARY)

cnt, hir = cv2.findContours(thr, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

ar = []

for c in cnt:

m = cv2.moments(c)

x = int(m["m10"]/m["m00"])

y = int(m["m01"]/m["m00"])

cv2.drawContours(img, cnt, -1, (0,0,255), 2)

cv2.circle(img, (x,y), 5, (255,0,0), -1)

a = cv2.contourArea(c)

ar.append(a)

ep = 0.1\*cv2.arcLength(c, True) # we can change 0.1 value to 0.01 and other to import the area of the rectangle

d = cv2.approxPolyDP(c, ep, True)

h = cv2.convexHull(d)

x,y,w,h = cv2.boundingRect(h)

cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 5)

print(ar)

cv2.imshow("Image", img)

cv2.imshow("Gray", gry)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Object Detections using Contours by Webcam

def nothing(x):

pass

cv2.namedWindow("moon")

cv2.createTrackbar("th", "moon", 0, 255, nothing)

cv2.createTrackbar("lb", "moon", 0, 255, nothing)

cv2.createTrackbar("lg", "moon", 0, 255, nothing)

cv2.createTrackbar("lr", "moon", 0, 255, nothing)

cv2.createTrackbar("ub", "moon", 255, 255, nothing)

cv2.createTrackbar("ug", "moon", 255, 255, nothing)

cv2.createTrackbar("ur", "moon", 255, 255, nothing)

cap = cv2.VideoCapture(0)

while cap.isOpened():

r, frame = cap.read()

if r == True:

thr = cv2.getTrackbarPos("th", "moon")

LB = cv2.getTrackbarPos("lb", "moon")

LG = cv2.getTrackbarPos("lg", "moon")

LR = cv2.getTrackbarPos("lr", "moon")

UB = cv2.getTrackbarPos("ub", "moon")

UG = cv2.getTrackbarPos("ug", "moon")

UR = cv2.getTrackbarPos("ur", "moon")

lower = np.array([LB, LG, LR])

upper = np.array([UB, UG, UR])

frame = cv2.resize(frame, (400,400))

frame = cv2.flip(frame, 1)

hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

mask = cv2.inRange(hsv, lower, upper)

res = cv2.bitwise\_and(frame, frame, mask = mask)

r, thi = cv2.threshold(mask, thr, 255, cv2.THRESH\_BINARY)

cnt, hr = cv2.findContours(thi, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cv2.drawContours(frame, cnt, -1, (255,0,0), 2)

cv2.imshow("thi", thi)

cv2.imshow("video", frame)

cv2.imshow("mask", mask)

cv2.imshow("res", res)

if cv2.waitKey(1) & 0xff == ord('q'):

break

else:

break

cap.release()

cv2.destroyAllWindows()

# Image Background Removal Histogram

img = cv2.imread("images\img\_5.jpeg")

hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

img1 = cv2.imread("images\img\_d5.jpg")

hsv1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2HSV)

hist = cv2.calcHist([hsv1], [0,1], None, [180, 256], [0,180,0,256])

mask = cv2.calcBackProject([hsv], [0,1], hist, [0,180,0,256],1)

ker = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5,5))

mask = cv2.filter2D(mask, -1, ker)

\_, thr = cv2.threshold(mask,250,255,cv2.THRESH\_BINARY)

mask = cv2.merge((mask, mask, mask))

res = cv2.bitwise\_or(img, mask)

cv2.imshow("res", res)

cv2.imshow("Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Hough Transformation Lines

## Method 1

img = cv2.imread("images\img\_19.png")

gry = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

edg = cv2.Canny(gry, 20, 250)

lines = cv2.HoughLines(edg, 1, np.pi/100,200)

for r, th in lines[0]:

a = np.cos(th)

b = np.sin(th)

x0 = a\*r

y0 = b\*r

x1 = int(x0+1000\*(-b))

y1 = int(y0+1000\*(a))

x2 = int(x0-1000\*(-b))

y2 = int(y0-1000\*(a))

cv2.line(img, (x1, y1), (x2, y2), (0,255,0),2)

cv2.imshow("Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

## Method 2

img = cv2.imread("images/img\_19.png")

gry = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

edg = cv2.Canny(gry, 20, 250)

l = cv2.HoughLinesP(edg, 1, np.pi/180, 200, minLineLength = 180, maxLineGap = 100)

for i in l:

x1, y1, x2, y2 = i[0]

cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 2)

cv2.imshow("Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Template Matching using opencv

img = cv2.imread("images\img\_20.jpeg")

gry = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

tem = cv2.imread("images\img\_20\_c.jpeg")

gry1 = cv2.cvtColor(tem, cv2.COLOR\_BGR2GRAY)

w, h = gry1.shape[::-1]

res = cv2.matchTemplate(gry, gry1, cv2.TM\_CCOEFF\_NORMED)

thr = 0.9

l = np.where(res>=thr)

for i in zip(\*l[::-1]):

cv2.rectangle(img, i, (i[0]+w,i[1]+h), (0,2550),2)

cv2.imshow("Image", img)

cv2.imshow("template", tem)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Hough Circle Detection

img = cv2.imread("images\img\_22.jpeg")

gry = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

gry = cv2.medianBlur(gry, 7)

c = cv2.HoughCircles(gry, cv2.HOUGH\_GRADIENT, 1, 20, param1=50, param2=50, minRadius=0, maxRadius=0)

data = np.uint16(np.around(c))

for (x,y,r) in data[0,:]:

cv2.circle(img, (x,y), r, (0,0,255), 2)

cv2.imshow("Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Grab cut Algorithm for background Change

img = cv2.imread("images/img\_7.jpg")

# Initialize mask for GrabCut

mask = np.zeros(img.shape[:2], np.uint8)

# Initialize background and foreground models

bgd\_model = np.zeros((1, 65), np.float64)

fgd\_model = np.zeros((1, 65), np.float64)

# Define the rectangle (x, y, width, height)

rect = (222, 23, 525 - 222, 407 - 23) # Format: (x, y, width, height)

# Apply GrabCut

cv2.grabCut(img, mask, rect, bgd\_model, fgd\_model, 10, cv2.GC\_INIT\_WITH\_RECT)

# Create a mask where probable background and definite background are set to 0

# and probable foreground and definite foreground are set to 1

mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype("uint8")

# Apply the mask to the image

img = img \* mask2[:, :, np.newaxis]

# Display the result

cv2.imshow("Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Video Background Removal using Algorithm

cap = cv2.VideoCapture(r"images\video\_2.mp4")

algo1 = cv2.createBackgroundSubtractorKNN() # detectShadows = True

algo2 = cv2.createBackgroundSubtractorMOG2() # detectShadows = True

while True:

r, frame = cap.read()

if r == True:

frame = cv2.resize(frame, (500,400))

r1 = algo1.apply(frame)

r2 = algo2.apply(frame)

cv2.imshow("Video", frame)

cv2.imshow("Algo1", r1)

cv2.imshow("Algo2", r2)

if cv2.waitKey(1) & 0xff == ord('p'):

break

else:

break

cap.release()

cv2.destroyAllWindows()

# Object Tracking and Detection

cap = cv2.VideoCapture(r"images\video\_2.mp4")

r, f = cap.read()

x, y, w, h = 300, 55, 250, 855 # Corrected height value

t = (x, y, w, h)

roi = f[y:y+h, x:x+w]

hsv\_roi = cv2.cvtColor(roi, cv2.COLOR\_BGR2HSV)

mask = cv2.inRange(hsv\_roi, np.array((0., 60., 32.)), np.array((180., 255., 255.)))

roi\_hist = cv2.calcHist([hsv\_roi], [0], mask, [180], [0, 180])

cv2.normalize(roi\_hist, roi\_hist, 0, 255, cv2.NORM\_MINMAX)

tr = (cv2.TERM\_CRITERIA\_EPS | cv2.TERM\_CRITERIA\_COUNT, 10, 1)

while True:

r, frame = cap.read()

if r == True:

hsv\_f = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

d = cv2.calcBackProject([hsv\_f], [0], roi\_hist, [0, 180], 1)

r, tp = cv2.meanShift(d, t, tr) #you can use cv2.CamShift instead of it

x, y, w, h = tp

final = cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 4)

cv2.imshow("Video", final)

if cv2.waitKey(1) & 0xFF == ord('p'):

break

else:

break

cap.release()

cv2.destroyAllWindows()

# Corner Detecting (Harris Corner Detection)