**Chapter 1 – Read Images Video Webcam**

## 1. **Read Image**

import cv2

img = cv2.imread("Resources/lena.png")

# DISPLAY

cv2.imshow("Lena Soderberg",img)

cv2.waitKey(0)

## 2. **Read Video**

import cv2

frameWidth = 640

frameHeight = 480

cap = cv2.VideoCapture("Resources/test\_ video.mp4")

while True:

success, img = cap.read()

img = cv2.resize(img, (frameWidth, frameHeight))

cv2.imshow("Result", img)

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

break

## 3. **Read Webcam**

import cv2

frameWidth = 640

frameHeight = 480

cap = cv2.VideoCapture(0)

cap.set(3, frameWidth)

cap.set(4, frameHeight)

cap.set(10, 150)

while True:

success, img = cap.read()

cv2.imshow("Result", img)

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

break

# Chapter 2 – Basic Functions

import cv2

import numpy as np

img = cv2.imread("Resources/lena.png")

kernel = np.ones((5,5),np.uint8)

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

imgBlur = cv2.GaussianBlur(imgGray,(7,7),0)

imgCanny = cv2.Canny(img,150,200)

imgDialation = cv2.dilate(imgCanny,kernel,iterations=1)

imgEroded = cv2.erode(imgDialation,kernel,iterations=1)

cv2.imshow("Gray Image",imgGray)

cv2.imshow("Blur Image",imgBlur)

cv2.imshow("Canny Image",imgCanny)

cv2.imshow("Dialation Image",imgDialation)

cv2.imshow("Eroded Image",imgEroded)

cv2.waitKey(0)

# Chapter 3 – Crop and Resize

import cv2

import numpy as np

img = cv2.imread("Resources/shapes.png")

print(img.shape)

imgResize = cv2.resize(img,(1000,500))

print(imgResize.shape)

imgCropped = img[46:119,352:495]

cv2.imshow("Image",img)

#cv2.imshow("Image Resize",imgResize)

cv2.imshow("Image Cropped",imgCropped)

cv2.waitKey(0)

# Chapter 4 – Shapes and Text

import cv2

import numpy as np

img = np.zeros((512,512,3),np.uint8)

#print(img)

#img[:]= 255,0,0

cv2.line(img,(0,0),(img.shape[1],img.shape[0]),(0,255,0),3)

cv2.rectangle(img,(0,0),(250,350),(0,0,255),2)

cv2.circle(img,(400,50),30,(255,255,0),5)

cv2.putText(img," OPENCV  ",(300,200),cv2.FONT\_HERSHEY\_COMPLEX,1,(0,150,0),3)

cv2.imshow("Image",img)

cv2.waitKey(0)

# Chapter 5 – Warp Perspective

import cv2

import numpy as np

img = cv2.imread("Resources/cards.jpg")

width,height = 250,350

pts1 = np.float32([[111,219],[287,188],[154,482],[352,440]])

pts2 = np.float32([[0,0],[width,0],[0,height],[width,height]])

matrix = cv2.getPerspectiveTransform(pts1,pts2)

imgOutput = cv2.warpPerspective(img,matrix,(width,height))

cv2.imshow("Image",img)

cv2.imshow("Output",imgOutput)

cv2.waitKey(0)

**multiple track bars with color detection**

import cv2

import numpy as np

def empty(a):

pass

cv2.namedWindow("TrackBars")

cv2.resizeWindow("TrackBars", 640, 240)

cv2.createTrackbar("Hue Min", "TrackBars", 0, 179, empty)

cv2.createTrackbar("Hue Max", "TrackBars", 19, 179, empty)

cv2.createTrackbar("Sat Min", "TrackBars", 110, 255, empty)

cv2.createTrackbar("Sat Max", "TrackBars", 240, 255, empty)

cv2.createTrackbar("Val Min", "TrackBars", 153, 255, empty)

cv2.createTrackbar("Val Max", "TrackBars", 255, 255, empty)

while True:

img = cv2.imread('E:\\opencv tutorial\\image\\lambo.png')

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

h\_min = cv2.getTrackbarPos("Hue Min", "TrackBars")

h\_max = cv2.getTrackbarPos("Hue Max", "TrackBars")

s\_min = cv2.getTrackbarPos("Sat Min", "TrackBars")

s\_max = cv2.getTrackbarPos("Sat Max", "TrackBars")

v\_min = cv2.getTrackbarPos("Val Min", "TrackBars")

v\_max = cv2.getTrackbarPos("Val Max", "TrackBars")

print(h\_min, h\_max, s\_min, s\_max, v\_min, v\_max)

lower = np.array([h\_min, s\_min, v\_min])

upper = np.array([h\_max, s\_max, v\_max])

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

imgResult = cv2.bitwise\_and(img, img, mask=mask)

cv2.imshow("Original",img)

cv2.imshow("HSV",imgHSV)

cv2.imshow("Mask", mask)

cv2.imshow("Result", imgResult)

cv2.waitKey(1)

# Chapter 6 – Joining Images

import cv2

import numpy as np

def stackImages(scale,imgArray):

rows = len(imgArray)

cols = len(imgArray[0])

rowsAvailable = isinstance(imgArray[0], list)

width = imgArray[0][0].shape[1]

height = imgArray[0][0].shape[0]

if rowsAvailable:

for x in range ( 0, rows):

for y in range(0, cols):

if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:

imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale, scale)

else:

imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0][0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)

if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor( imgArray[x][y], cv2.COLOR\_GRAY2BGR)

imageBlank = np.zeros((height, width, 3), np.uint8)

hor = [imageBlank]\*rows

hor\_con = [imageBlank]\*rows

for x in range(0, rows):

hor[x] = np.hstack(imgArray[x])

ver = np.vstack(hor)

else:

for x in range(0, rows):

if imgArray[x].shape[:2] == imgArray[0].shape[:2]:

imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)

else:

imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0]), None,scale, scale)

if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR\_GRAY2BGR)

hor= np.hstack(imgArray)

ver = hor

return ver

img = cv2.imread('Resources/lena.png')

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

imgStack = stackImages(0.5,([img,imgGray,img],[img,img,img]))

# imgHor = np.hstack((img,img))

# imgVer = np.vstack((img,img))

#

# cv2.imshow("Horizontal",imgHor)

# cv2.imshow("Vertical",imgVer)

cv2.imshow("ImageStack",imgStack)

cv2.waitKey(0)

# Chapter 8 – Contour/Shape Detection

**# 1 shape detection**

import cv2

img = cv2.imread('E:\\opencv tutorial\\image\\shapes.png')

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

ret, thrash = cv2.threshold(imgGry, 240, 255, cv2.CHAIN\_APPROX\_NONE)

contours, hierarchy = cv2.findContours(thrash, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)

for contour in contours:

approx = cv2.approxPolyDP(contour, 0.01 \* cv2.arcLength(contour, True), True)

cv2.drawContours(img, [approx], 0, (0, 0, 0), 5)

x = approx.ravel()[0]

y = approx.ravel()[1] - 5

if len(approx) == 3:

cv2.putText(img, "Triangle", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

elif len(approx) == 4:

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

aspectRatio = float(w) / h

print(aspectRatio)

if aspectRatio >= 0.95 and aspectRatio < 1.05:

cv2.putText(img, "square", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

else:

cv2.putText(img, "rectangle", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

elif len(approx) == 5:

cv2.putText(img, "pentagon", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

elif len(approx) == 10:

cv2.putText(img, "star", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

else:

cv2.putText(img, "circle", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

cv2.imshow('shapes', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**# 2 shape detection from murtaza workshop**

import cv2

import numpy as np

def getContours(img):

contours,hierarchy = cv2.findContours(img,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_NONE)

for cnt in contours:

area = cv2.contourArea(cnt)

print(area)

if area>500:

cv2.drawContours(imgContour, cnt, -1, (255, 0, 0), 3)

peri = cv2.arcLength(cnt,True)

#print(peri)

approx = cv2.approxPolyDP(cnt,0.02\*peri,True)

print(len(approx))

objCor = len(approx)

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

if objCor ==3: objectType ="Tri"

elif objCor == 4:

aspRatio = w/float(h)

if aspRatio >0.98 and aspRatio <1.03: objectType= "Square"

else:objectType="Rectangle"

elif objCor>4: objectType= "Circles"

else:objectType="None"

cv2.rectangle(imgContour,(x,y),(x+w,y+h),(0,255,0),2)

cv2.putText(imgContour,objectType,

(x+(w//2)-10,y+(h//2)-10),cv2.FONT\_HERSHEY\_COMPLEX,0.7,

(0,0,0),2)

img = cv2.imread('E:\\opencv tutorial\\image\\shapes.png')

imgContour = img.copy()

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

imgBlur = cv2.GaussianBlur(imgGray,(7,7),1)

imgCanny = cv2.Canny(imgBlur,50,50)

getContours(imgCanny)

cv2.imshow('result image', imgContour)

cv2.waitKey(0)

# Chapter 9 – Face Detection

import cv2

faceCascade= cv2.CascadeClassifier("E:opencv tutorial\cascadefile\haarcascade\_frontalface\_default.xml")

img = cv2.imread('E:\opencv tutorial\lena.png')

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

faces = faceCascade.detectMultiScale(imgGray,1.1,4)

for (x,y,w,h) in faces:

cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

cv2.imshow("Result", img)

cv2.waitKey(0)

**Faces and eyes detect**

*''''  
# face and eye detect for image  
import cv2  
  
img = cv2.imread('E:\opencv tutorial\lena.png')  
face\_cascade = cv2.CascadeClassifier("E:\opencv tutorial\cascade file\haarcascade\_frontalface\_default.xml")  
eye\_cascade = cv2.CascadeClassifier("E:\opencv tutorial\cascade file\haarcascade\_eye.xml")  
  
faces = face\_cascade.detectMultiScale(img, 1.1, 4)  
  
for (x, y, w, h) in faces:  
 cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0),2)  
 eyes = eye\_cascade.detectMultiScale(img, 1.1, 3)  
 for (ex, ey, ew, eh) in eyes:  
 cv2.rectangle(img, (ex, ey), (ex + ew, ey + eh), (0, 255, 0),2)  
cv2.imshow('image', img)  
cv2.waitKey(0)  
'''*'''# 2 face and eye detect for webcam  
  
import cv2  
  
cap = cv2.VideoCapture(0)  
face\_cascade = cv2.CascadeClassifier("E:\opencv tutorial\cascade file\haarcascade\_frontalface\_default.xml")  
eye\_cascade = cv2.CascadeClassifier("E:\opencv tutorial\cascade file\haarcascade\_eye.xml")  
  
while True:  
 success, img = cap.read(0)  
 faces = face\_cascade.detectMultiScale(img, 1.1, 4)  
 for (x, y, w, h) in faces:  
 cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)  
 eyes = eye\_cascade.detectMultiScale(img, 1.1, 3)  
 for (ex, ey, ew, eh) in eyes:  
 cv2.rectangle(img, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 2)  
 cv2.imshow('image', img)  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
'''  
  
# 3 face and eye detect for webcam

import cv2

cap = cv2.VideoCapture("E:\\opencv tutorial\\baba.mp4")  
face\_cascade = cv2.CascadeClassifier("E:\opencv tutorial\cascade file\haarcascade\_frontalface\_default.xml")  
eye\_cascade = cv2.CascadeClassifier("E:\opencv tutorial\cascade file\haarcascade\_eye.xml")  
  
while True:  
 success, img = cap.read(0)  
 faces = face\_cascade.detectMultiScale(img, 1.1, 4)  
 for (x, y, w, h) in faces:  
 cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)  
 eyes = eye\_cascade.detectMultiScale(img, 1.1, 3)  
 for (ex, ey, ew, eh) in eyes:  
 cv2.rectangle(img, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 2)  
 cv2.imshow('image', img)  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break

# Project 1 – Virtual Paint

import cv2

import numpy as np

frameWidth = 640

frameHeight = 480

cap = cv2.VideoCapture(0)

cap.set(3, frameWidth)

cap.set(4, frameHeight)

cap.set(10,150)

myColors = [[5,107,0,19,255,255],

[133,56,0,159,156,255],

[57,76,0,100,255,255],

[90,48,0,118,255,255]]

myColorValues = [[51,153,255], ## BGR

[255,0,255],

[0,255,0],

[255,0,0]]

myPoints = [] ## [x , y , colorId ]

def findColor(img,myColors,myColorValues):

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

count = 0

newPoints=[]

for color in myColors:

lower = np.array(color[0:3])

upper = np.array(color[3:6])

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

x,y=getContours(mask)

cv2.circle(imgResult,(x,y),15,myColorValues[count],cv2.FILLED)

if x!=0 and y!=0:

newPoints.append([x,y,count])

count +=1

#cv2.imshow(str(color[0]),mask)

return newPoints

def getContours(img):

contours,hierarchy = cv2.findContours(img,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_NONE)

x,y,w,h = 0,0,0,0

for cnt in contours:

area = cv2.contourArea(cnt)

if area>500:

#cv2.drawContours(imgResult, cnt, -1, (255, 0, 0), 3)

peri = cv2.arcLength(cnt,True)

approx = cv2.approxPolyDP(cnt,0.02\*peri,True)

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

return x+w//2,y

def drawOnCanvas(myPoints,myColorValues):

for point in myPoints:

cv2.circle(imgResult, (point[0], point[1]), 10, myColorValues[point[2]], cv2.FILLED)

while True:

success, img = cap.read()

imgResult = img.copy()

newPoints = findColor(img, myColors,myColorValues)

if len(newPoints)!=0:

for newP in newPoints:

myPoints.append(newP)

if len(myPoints)!=0:

drawOnCanvas(myPoints,myColorValues)

cv2.imshow("Result", imgResult)

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

break

## **Color Picker**

import cv2

import numpy as np

frameWidth = 640

frameHeight = 480

cap = cv2.VideoCapture(0)

cap.set(3, frameWidth)

cap.set(4, frameHeight)

def empty(a):

pass

cv2.namedWindow("HSV")

cv2.resizeWindow("HSV", 640, 240)

cv2.createTrackbar("HUE Min", "HSV", 0, 179, empty)

cv2.createTrackbar("HUE Max", "HSV", 179, 179, empty)

cv2.createTrackbar("SAT Min", "HSV", 0, 255, empty)

cv2.createTrackbar("SAT Max", "HSV", 255, 255, empty)

cv2.createTrackbar("VALUE Min", "HSV", 0, 255, empty)

cv2.createTrackbar("VALUE Max", "HSV", 255, 255, empty)

while True:

success, img = cap.read()

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

h\_min = cv2.getTrackbarPos("HUE Min", "HSV")

h\_max = cv2.getTrackbarPos("HUE Max", "HSV")

s\_min = cv2.getTrackbarPos("SAT Min", "HSV")

s\_max = cv2.getTrackbarPos("SAT Max", "HSV")

v\_min = cv2.getTrackbarPos("VALUE Min", "HSV")

v\_max = cv2.getTrackbarPos("VALUE Max", "HSV")

print(h\_min)

lower = np.array([h\_min, s\_min, v\_min])

upper = np.array([h\_max, s\_max, v\_max])

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

result = cv2.bitwise\_and(img, img, mask=mask)

mask = cv2.cvtColor(mask, cv2.COLOR\_GRAY2BGR)

hStack = np.hstack([img, mask, result])

cv2.imshow('Horizontal Stacking', hStack)

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

break

cap.release()

cv2.destroyAllWindows()

# Project 2 – Document Scanner

import cv2

import numpy as np

###################################

widthImg=540

heightImg =640

#####################################

cap = cv2.VideoCapture(0)

cap.set(10,150)

def preProcessing(img):

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

imgBlur = cv2.GaussianBlur(imgGray,(5,5),1)

imgCanny = cv2.Canny(imgBlur,200,200)

kernel = np.ones((5,5))

imgDial = cv2.dilate(imgCanny,kernel,iterations=2)

imgThres = cv2.erode(imgDial,kernel,iterations=1)

return imgThres

def getContours(img):

biggest = np.array([])

maxArea = 0

contours,hierarchy = cv2.findContours(img,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_NONE)

for cnt in contours:

area = cv2.contourArea(cnt)

if area>5000:

#cv2.drawContours(imgContour, cnt, -1, (255, 0, 0), 3)

peri = cv2.arcLength(cnt,True)

approx = cv2.approxPolyDP(cnt,0.02\*peri,True)

if area >maxArea and len(approx) == 4:

biggest = approx

maxArea = area

cv2.drawContours(imgContour, biggest, -1, (255, 0, 0), 20)

return biggest

def reorder (myPoints):

myPoints = myPoints.reshape((4,2))

myPointsNew = np.zeros((4,1,2),np.int32)

add = myPoints.sum(1)

#print("add", add)

myPointsNew[0] = myPoints[np.argmin(add)]

myPointsNew[3] = myPoints[np.argmax(add)]

diff = np.diff(myPoints,axis=1)

myPointsNew[1]= myPoints[np.argmin(diff)]

myPointsNew[2] = myPoints[np.argmax(diff)]

#print("NewPoints",myPointsNew)

return myPointsNew

def getWarp(img,biggest):

biggest = reorder(biggest)

pts1 = np.float32(biggest)

pts2 = np.float32([[0, 0], [widthImg, 0], [0, heightImg], [widthImg, heightImg]])

matrix = cv2.getPerspectiveTransform(pts1, pts2)

imgOutput = cv2.warpPerspective(img, matrix, (widthImg, heightImg))

imgCropped = imgOutput[20:imgOutput.shape[0]-20,20:imgOutput.shape[1]-20]

imgCropped = cv2.resize(imgCropped,(widthImg,heightImg))

return imgCropped

def stackImages(scale,imgArray):

rows = len(imgArray)

cols = len(imgArray[0])

rowsAvailable = isinstance(imgArray[0], list)

width = imgArray[0][0].shape[1]

height = imgArray[0][0].shape[0]

if rowsAvailable:

for x in range ( 0, rows):

for y in range(0, cols):

if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:

imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale, scale)

else:

imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0][0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)

if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor( imgArray[x][y], cv2.COLOR\_GRAY2BGR)

imageBlank = np.zeros((height, width, 3), np.uint8)

hor = [imageBlank]\*rows

hor\_con = [imageBlank]\*rows

for x in range(0, rows):

hor[x] = np.hstack(imgArray[x])

ver = np.vstack(hor)

else:

for x in range(0, rows):

if imgArray[x].shape[:2] == imgArray[0].shape[:2]:

imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)

else:

imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0]), None,scale, scale)

if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR\_GRAY2BGR)

hor= np.hstack(imgArray)

ver = hor

return ver

while True:

success, img = cap.read()

img = cv2.resize(img,(widthImg,heightImg))

imgContour = img.copy()

imgThres = preProcessing(img)

biggest = getContours(imgThres)

if biggest.size !=0:

imgWarped=getWarp(img,biggest)

# imageArray = ([img,imgThres],

# [imgContour,imgWarped])

imageArray = ([imgContour, imgWarped])

cv2.imshow("ImageWarped", imgWarped)

else:

# imageArray = ([img, imgThres],

# [img, img])

imageArray = ([imgContour, img])

stackedImages = stackImages(0.6,imageArray)

cv2.imshow("WorkFlow", stackedImages)

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

break