NITTE (Deemed to be University) | NMAM INSTITUTE OF TECHNOLOGY

Nitte(DU) established under Section 3 of UGC Act 1956 | Accredited with 'A+' Grade by NAAC

Department of Artificial Intelligence & Machine Learning Engineering

LAB MANUAL

Pattern and Visual Recognition Lab 20AM507

Academic Year

2022-2023

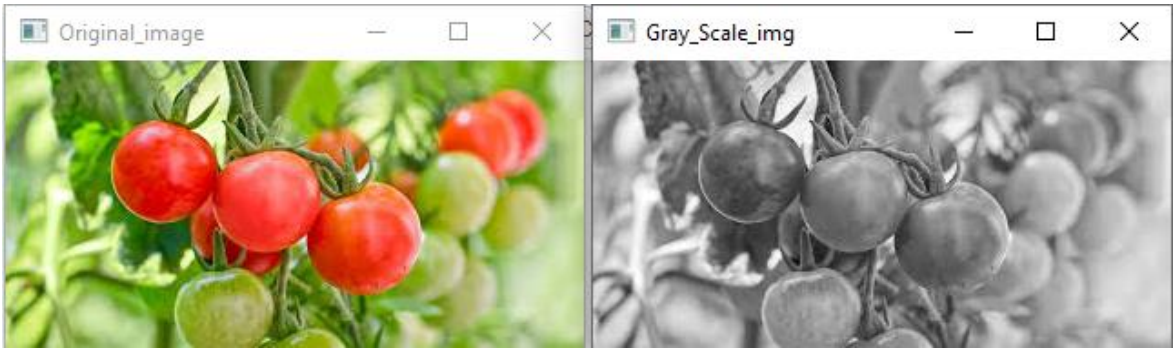
PATTERN AND VISUAL RECOGNITION LAB		
Course Code	20AM507	**CIE Marks**
Number of Contact Hours/Week	0:0:3	**SEE Marks**
Total Number of Contact Hours	39	**Exam Hours**

1. **Working with images and videos in OpenCV**
2. **Bitwise Operations on Binary Images**
3. **Draw geometric shapes on images using OpenCV**
4. **Morphological operations based on OpenCV**
5. **Thresholding, Edge detection and Contour detection**
6. **Opencv Python program for Face and Eye Detection**
7. **YOLO object detection using OpenCV**
8. **Handwritten Digit Recognition on MNIST dataset**

Working with images and videos in OpenCV

```
import cv2
import matplotlib.pyplot as plt

# To read image from disk, we use
# cv2.imread function, in below method,
img = cv2.imread("tomato.jpg", cv2.IMREAD_COLOR)

# Converting BGR color to RGB color format
RGB_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

#Displaying image using plt.imshow() method
plt.imshow(RGB_img)

# Using cv2.imread() method
# Using 0 to read image in grayscale mode
Gray_img = cv2.imread('tomato.jpg', 0)

# Filename
filename ='Gray_image.jpg'

# Using cv2.imwrite() method
# Saving the image
cv2.imwrite(filename, Gray_img)

# Creating GUI window to display an image on screen
# first Parameter is windows title (should be in string format)
# Second Parameter is image array
cv2.imshow("Original_image", img)

# Displaying the image
cv2.imshow('Gray_Scale_img', Gray_img)

# To hold the window on screen, we use cv2.waitKey method
# Once it detected the close input, it will release the control
# To the next line
# First Parameter is for holding screen for specified milliseconds
# It should be positive integer. If 0 pass an parameter, then it will
# hold the screen until user close it.
cv2.waitKey(0)

# It is for removing/deleting created GUI window from screen
# and memory
cv2.destroyAllWindows()

# Create a VideoCapture object and read from input file
cap = cv2.VideoCapture('Sample.mp4')
# Check if camera opened successfully
if (cap.isOpened()== False):
```

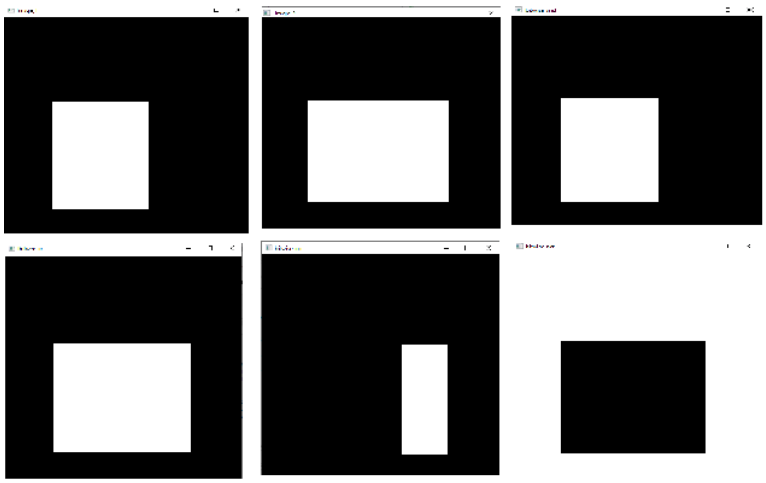
```
    print("Error opening video file")

# Read until video is completed
while(cap.isOpened()):
# Capture frame-by-frame
    ret, frame = cap.read()
    if ret == True:
    # Display the resulting frame
        cv2.imshow('Frame', frame)

    # Press Q on keyboard to exit
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break

# Break the loop
    else:
        break

# When everything done, release
# the video capture object
cap.release()
cv2.destroyAllWindows()
```



Bitwise Operations on Binary Images

```
import numpy as np
import cv2

# Creating a black screen image using nupy.zeros function
Img_1 = np.zeros((512, 512, 3), dtype='uint8')
Img_2 = np.zeros((512, 512, 3), dtype='uint8')
Img_1 = cv2.rectangle(Img_1, (100,200), (300,450),(255,255,255), -1)
Img_2 = cv2.rectangle(Img_2, (100,200), (400,450),(255,255,255), -1)
bitwise_and = cv2.bitwise_and(Img_2, Img_1, mask=None)
bitwise_or = cv2.bitwise_or(Img_2, Img_1, mask=None)
bitwise_xor = cv2.bitwise_xor(Img_2, Img_1, mask=None)
bitwise_not =cv2.bitwise_not(Img_2, mask=None)
cv2.imshow('Image_1', Img_1)
```

```
cv2.imshow('Image_2', Img_2)
cv2.imshow('bitwise_and', bitwise_and)
cv2.imshow('bitwise_or', bitwise_or)
cv2.imshow('bitwise_xor', bitwise_xor)
cv2.imshow('bitwise_not', bitwise_not)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Draw geometric shapes on images using OpenCV

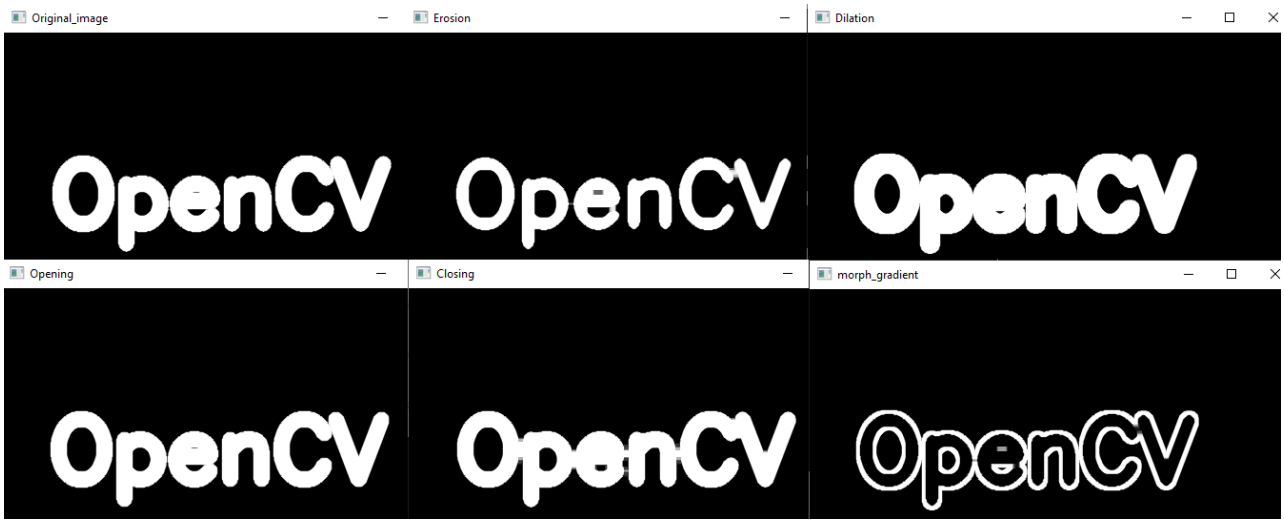
```
import numpy as np
import cv2

# Creating a black screen image using nupy.zeros function
Img = np.zeros((512, 512, 3), dtype='uint8')

# Using cv2.line() method to draw a diagonal white line with thickness of 4
px
# Start coordinate, here (0, 0). It represents the top left corner of image
start_point = (0, 0)
# End coordinate, here (100, 100). It represents the bottom right corner of
the image according to resolution
end_point = (100, 100)
# White color in BGR
color = (255, 255, 255)
# Line thickness of 4 px
thickness = 4
Img =cv2.line(Img, start_point, end_point, color, thickness)

# Using cv2.arrowedLine() method Draw a diagonal arrow line
# with thickness of 4 px
start_point = (0, 50)
end_point = (100, 150)
color = (255, 255, 255)
thickness = 4
Img = cv2.arrowedLine(Img, start_point, end_point,color, thickness)
```

```
# Using cv2.ellipse() method
# Draw a ellipse with blue line borders of thickness of -1 px
center_coordinates = (300, 100)
axesLength = (100, 50)
angle = 30
startAngle = 0
endAngle = 360
color = (255, 0, 0)
thickness = -1
Img = cv2.ellipse(Img, center_coordinates, axesLength, angle,startAngle,
endAngle, color, thickness)

# Using cv2.circle() method
# Draw a circle with blue line borders of thickness of 4 px
center_coordinates = (450, 100)
radius = 30
color = (255, 0, 0)
thickness = 4
Img =cv2.circle(Img, center_coordinates, radius, color, thickness)

# represents the top left corner of rectangle
start_point = (20, 200)
# represents the bottom right corner of rectangle
end_point = (200, 300)
color = (0, 255, 0)
thickness = 2
# Using cv2.rectangle() method
# Draw a rectangle with green line borders of thickness of 2 px
Img = cv2.rectangle(Img, start_point, end_point, color, thickness)

# Using cv2.putText() method
font = cv2.FONT_HERSHEY_SIMPLEX
org = (50, 400)
fontScale = 2
color = (0, 0, 255)
thickness = 3
Img = cv2.putText(Img, 'OpenCV', org, font,fontScale, color,
thickness,cv2.LINE_AA, False)
Img = cv2.putText(Img, 'OpenCV', org, font,fontScale, color,
thickness,cv2.LINE_AA, True)

# Displaying the image
cv2.imshow('Geometric_shapes', Img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Morphological operations based on OpenCV

```
import cv2
import numpy as np

Img = np.zeros((512, 512, 3), dtype='uint8')

# Using cv2.putText() method
font = cv2.FONT_HERSHEY_SIMPLEX
org = (50,200)
fontScale = 3
color = (255, 255, 255)
thickness = 15
Img = cv2.putText(Img, 'OpenCV', org, font,fontScale, color,
thickness,cv2.LINE_AA)

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

# Using cv2.erode() method
img_erosion =cv2.erode(Img, kernel, iterations=1)
# Using cv2.dilate() method
img_dilation =cv2.dilate(Img, kernel, iterations=1)
# opening the image
img_opening = cv2.morphologyEx(Img, cv2.MORPH_OPEN, kernel, iterations=1)
# closing the image
img_closing = cv2.morphologyEx(Img, cv2.MORPH_CLOSE, kernel, iterations=1)
# use morph gradient
img_morph_gradient = cv2.morphologyEx(Img, cv2.MORPH_GRADIENT, kernel)

# Displaying the image
cv2.imshow('Original_image', Img)
cv2.imshow('Erosion', img_erosion)
cv2.imshow('Dilation', img_dilation)
cv2.imshow('Opening', img_opening)
```

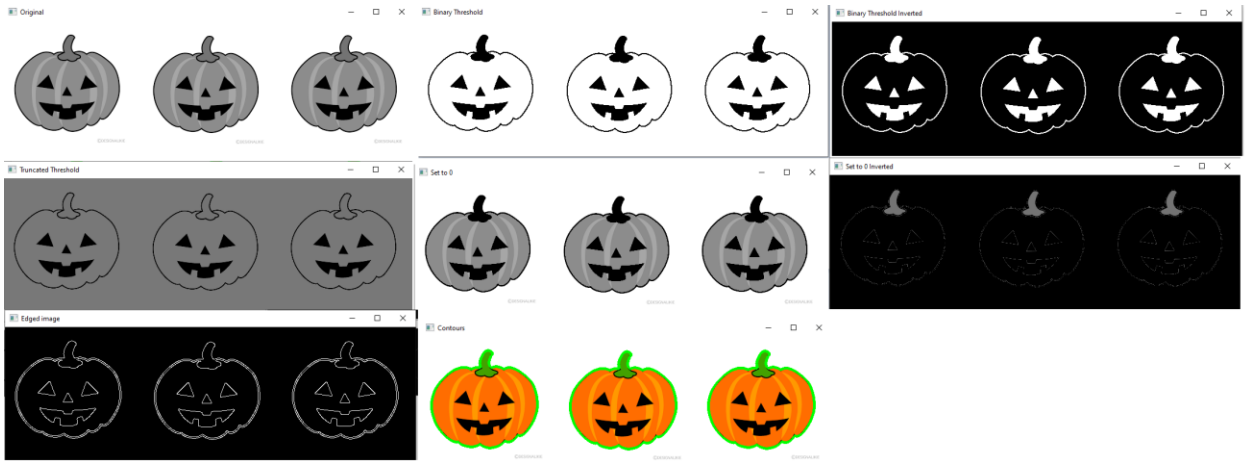
```
cv2.imshow('Closing', img_closing)
cv2.imshow('morph_gradient', img_morph_gradient)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



Thresholding, Edge detection and Contour detection

```
# Python program to illustrate
# simplethresholding type on an image
import cv2
import numpy as np
from matplotlib import pyplot as plt

image1 =cv2.imread('hallowen.png')
img =cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

# applying different thresholding
# techniques on the input image
#cv2.THRESH_BINARY: If pixel intensity is greater than
#the set threshold, value set to 255, else set to 0 (black).
ret, thresh1 =cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)
#cv2.THRESH_BINARY_INV: Inverted or Opposite case of cv2.THRESH_BINARY.
ret, thresh2 =cv2.threshold(img, 120, 255, cv2.THRESH_BINARY_INV)
#cv.THRESH_TRUNC: If pixel intensity value is greater than threshold,
#it is truncated to the threshold. The pixel values are set to be the
#same as the threshold. All other values remain the same.
ret, thresh3 =cv2.threshold(img, 120, 255, cv2.THRESH_TRUNC)
#cv.THRESH_TOZERO: Pixel intensity is set to 0, for all the pixels
#intensity, less than the threshold value.
ret, thresh4 =cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO)
#cv.THRESH_TOZERO_INV: Inverted or Opposite case of cv2.THRESH_TOZERO.
ret, thresh5 =cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO_INV)
```

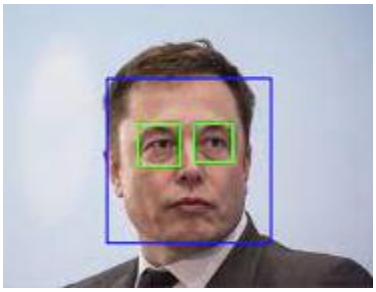
```
# the window showing output images
# with the corresponding thresholding
# techniques applied to the input images
cv2.imshow('Original', img)
cv2.imshow('Binary Threshold', thresh1)
cv2.imshow('Binary Threshold Inverted', thresh2)
cv2.imshow('Truncated Threshold', thresh3)
cv2.imshow('Set to 0', thresh4)
cv2.imshow('Set to 0 Inverted', thresh5)

blurred = cv2.GaussianBlur(thresh1, (3, 3), 0)
edged = cv2.Canny(blurred, 10, 100)
cv2.imshow("Edged image", edged)

# Finding Contours
# Use a copy of the image e.g. edged.copy()
# since findContours alters the image
contours, hierarchy = cv2.findContours(edged,cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
print("Number of Contours found = " + str(len(contours)))

# Draw all contours
# -1 signifies drawing all contours
cv2.drawContours(image1, contours, -1, (0, 255, 0), 3)
cv2.imshow('Contours', image1)
# De-allocate any associated memory usage
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Opencv Python program for Face and Eye Detection

```
# OpenCV program to detect face in real time
import cv2

# load the required trained XML classifiers
# https://github.com/opencv/opencv/tree/master/data/haarcascades
# object we want to detect a cascade function is trained
```

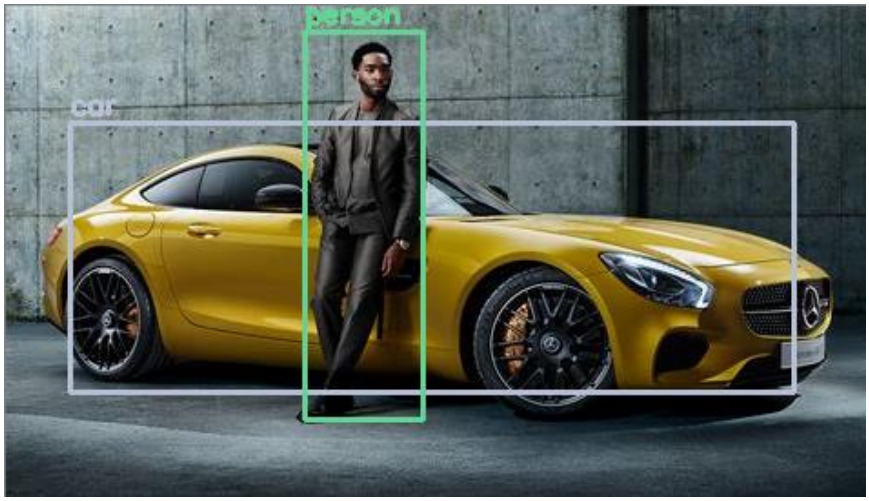
```
# from a lot of positive(faces) and negative(non-faces) images.
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# https://github.com/opencv/opencv/tree/master/data/haarcascades
# Trained XML file for detecting eyes
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

# capture frames from a camera
cap = cv2.VideoCapture(0)

# loop runs if capturing has been initialized.
while 1:

    # reads frames from a camera
    ret, img = cap.read()

    # convert to gray scale of each frames
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Detects faces of different sizes in the input image
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:
        # To draw a rectangle in a face
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]

        # Detects eyes of different sizes in the input image
        eyes = eye_cascade.detectMultiScale(roi_gray)

        #To draw a rectangle in eyes
        for (ex,ey,ew,eh) in eyes:
            cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

    # Display an image in a window
    cv2.imshow('img',img)

    # Wait for Esc key to stop
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

# Close the window
cap.release()

# De-allocate any associated memory usage
cv2.destroyAllWindows()
```



YOLO object detection using OpenCV

```
import cv2
import numpy as np
# Load Yolo
print("LOADING YOLO")
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
#save all the names in file o the list classes
classes = []
with open("coco.names", "r") as f:
      classes = [line.strip() for line in f.readlines()]
#get layers of the network
layer_names = net.getLayerNames()
#Determine the output layer names from the YOLO model
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
print("YOLO LOADED")
# Capture frame-by-frame
img = cv2.imread("test_img.jpg")
#img = cv2.resize(img, None, fx=0.4, fy=0.4)
height, width, channels = img.shape
# USing blob function of opencv to preprocess image
blob = cv2.dnn.blobFromImage(img, 1 / 255.0, (416, 416),swapRB=True,
crop=False)
#Detecting objects
net.setInput(blob)
outs = net.forward(output_layers)
# Showing informations on the screen
class_ids = []
confidences = []
boxes = []
for out in outs:
      for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5:
                  # Object detected
                  center_x = int(detection[0] * width)
                  center_y = int(detection[1] * height)
                  w = int(detection[2] * width)
```

```
                    h = int(detection[3] * height)
                    # Rectangle coordinates
                    x = int(center_x - w / 2)
                    y = int(center_y - h / 2)
                    boxes.append([x, y, w, h])
                    confidences.append(float(confidence))
                    class_ids.append(class_id)
#We use NMS function in opencv to perform Non-maximum Suppression
#we give it score threshold and nms threshold as arguments.
indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
colors = np.random.uniform(0, 255, size=(len(classes), 3))
for i in range(len(boxes)):
      if i in indexes:
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])

            color = colors[class_ids[i]]
            cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
            cv2.putText(img, label, (x, y -5),cv2.FONT_HERSHEY_SIMPLEX,1/2,
            color, 2)
cv2.imshow("Image",img)
cv2.waitKey(0)
```



Handwritten Digit Recognition on MNIST dataset

```
# fetching dataset
from sklearn.datasets import fetch_openml
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
mnist = fetch_openml('mnist_784')
x, y = mnist['data'], mnist['target']
some_digit = x.to_numpy()[36001]
```

```
some_digit_image = some_digit.reshape(28, 28) # let's reshape to plot it
plt.imshow(some_digit_image,
cmap=matplotlib.cm.binary,interpolation='nearest')
plt.axis("off")
plt.show()
x_train, x_test = x[:60000], x[60000:70000]
y_train, y_test = y[:60000], y[60000:70000]
shuffle_index = np.random.permutation(60000)
#x_train, y_train = x_train.[shuffle_index], y_train.[shuffle_index]
# Creating a 2-detector
y_train = y_train.astype(np.int8)
y_test = y_test.astype(np.int8)
y_train_2 = (y_train == 2)
y_test_2 = (y_test == 2)
# Train a logistic regression classifier
clf = LogisticRegression(tol=0.1)
# Train a logistic regression classifier
clf = LogisticRegression(tol=0.1)
clf.fit(x_train, y_train_2)
example = clf.predict([some_digit])
print(example)
# Cross Validation
a = cross_val_score(clf, x_train, y_train_2, cv=3, scoring="accuracy")
print(a.mean())
```