Task

On

OpenCv

**Course**: Artificial Intelligence

(Machine Learning & Deep Learning)

Name: Abbas Shafi

Father Name: Dr Muhammad Shafi

Week: 19

Submitted to: Sir Syed Nazir Afridi



National Vocational & Technical Training Commission

National Center For Big Data & Cloud Computing

University of Engineering & Technology

Peshawar

[ ]:

[ ]:

*# Load our input image*

image=cv2.imread('E:\Speech to Text\Hasan-Ali.jpg') cv2.imshow('Original',image)

cv2.waitKey()

*# We use cvtColor, to convert to grayscale*

gray = cv2.cvtColor(image , cv2.COLOR\_BGR2GRAY) cv2.imshow('GrayScale', gray)

cv2.waitKey() cv2.destroyAllWindows()

# GrayScale Image

**import cv2**

[ ]:

**import numpy as np**

image = cv2.imread('E:\Speech to Text\Hasan-Ali.jpg',0) height, width = image.shape

*# Extract Sobel Edges*

sobel\_x = cv2.Sobel(image, cv2.CV\_64F, 0, 1, ksize=5) sobel\_y = cv2.Sobel(image, cv2.CV\_64F, 1, 0, ksize=5)

cv2.imshow('Original', image) cv2.waitKey(0) cv2.imshow('Sobel X', sobel\_x) cv2.waitKey(0) cv2.imshow('Sobel Y', sobel\_y) cv2.waitKey(0)

sobel\_OR = cv2.bitwise\_or(sobeal\_x, sobel\_y)

* 1. **Edge Detection & Image Gradients**

cv2.imshow('sobel\_OR', sobel\_OR) cv2.waitKey(0)

laplacian = cv2.Laplacian(image, cv2.CV\_64F) cv2.imshow('Laplacian', laplacian) cv2.waitKey(0)

*## Then, we need to provide two values: threshold1 and threshold2. Any*␣

*‹→gradient value larger than threshold2*

*# is considered to be an edge. Any value below threshold1 is considered not to*␣

*‹→be an edge.*

*#Values in between threshold1 and threshold2 are either classi ed as edges or*␣

*‹→non-edges based on how their*

*#intensities are “connected”. In this case, any gradient values below 60 are*␣

*‹→considered non-edges*

*#whereas any values above 120 are considered edges.*

[ ]:

# define a video capture object

vid = cv2.VideoCapture(0)

**while**(**True**):

# Executes without any condition until break

# Capture the video frame # by frame

ret , frame = vid.read()

# Display the resulting frame

cv2.imshow('frame', frame)

# the 'q' button is set as the

# quitting button you may use any # desired button of your choice

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

**break**

*# Canny Edge Detection uses gradient values as thresholds # The first threshold gradient*

canny = cv2.Canny(image, 50, 120) cv2.imshow('Canny', canny) cv2.waitKey(0)

cv2.destroyAllWindows()

# Live Video Capturing

#After the loop release the cap object

vid.release()

# Destroy all the windows

cv2.destroyAllWindows()

# Closes video files or capturing device

[ ]:

**import numpy as np**

*# Create a VideoCapture object and read from input file*

*# If the input is the camera, pass 0 instead of the video file name*

cap = cv2.VideoCapture('D:\AI and DL Course\deep learning course\Master␣

*‹→*OpenCV\imagecv**\\**walking.avi')

*# Check if camera opened successfully*

**if** (cap.isOpened()== **False**):

print("Error opening video stream or 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()

*# Closes all the frames*

cv2.destroyAllWindows()