**APPENDIX B**

**OBJECTS DETECTION AND CLASSIFICATION PROGRAM FOR LANE1**

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Mon Mar 23 23:00:54 2020

@author: AungMinKhant

"""

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import mysql.connector

from mysql.connector import Error

import threading

import cv2

import numpy as np

import vehicles

import time

loop = 0

idnum = 0

rowcount = 0

exitFlag = 0

delay = 0

flow = 0

saturation = 0

first\_id = 0

def UpdateLane(fid,flow,sat):

rowcount = 1

first\_id = 0

try:

connection = mysql.connector.connect(host='localhost',database='traffic',user='minkhant',password='root')

if connection.is\_connected():

cursor = connection.cursor()

cursor.execute("select database();")

recoord = cursor.fetchone()

query = """UPDATE lane1 SET flowrate = %s, saturation = %s WHERE id = %s"""

print("Flow rate ",flow)

print("Saturatioin Flow ",sat)

print("Update ID -",fid)

data = (flow,sat,fid)

cursor.execute(query,data)

connection.commit()

except Error as e:

print("Error While connecting to MySQL")

finally:

if(connection.is\_connected()):

cursor.close()

connection.close()

def CountJam(flow,fid,loop):

try:

if(loop == 1):

numcar = 0

# net = cv2.dnn.readNet("yolov3-tiny.weights","yolov3-tiny.cfg")

net = cv2.dnn.readNet("yolov3.weights","yolov3.cfg")

classes = []

with open("coco.names","r") as f:

classes = [line.strip() for line in f.readlines()]

layer\_names = net.getLayerNames()

outputlayers = [layer\_names[i[0]-1] for i in net.getUnconnectedOutLayers()]

colors = np.random.uniform(0, 255, size=(len(classes), 3))

#loading image

img = cv2.imread("car.jpg")

height, width, channels = img.shape

blob = cv2.dnn.blobFromImage(img,0.00392,(416,416),(0,0,0),True,crop=False)

net.setInput(blob)

outs = net.forward(outputlayers)

#showing information 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

centerX = int(detection[0] \* width)

centerY = int(detection[1] \* height)

w = int(detection[2] \* width)

h = int(detection[3] \* height)

#Draw Rectangle

x = int(centerX - w / 2)

y = int(centerY - h / 2)

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

boxes.append([x, y, w, h])

confidences.append(float(confidence))

class\_ids.append(class\_id)

# print(centerX)

# print(centerY)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.3, 0.4)

font = cv2.FONT\_HERSHEY\_SIMPLEX

for i in range(len(boxes)):

if i in indexes:

x, y, w, h = boxes[i]

label = classes[class\_ids[i]]

color = colors[i]

if label == 'car' or label == 'truck' or label == 'bus':

numcar += 1

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

cv2.putText(img, label, (x, y + 10), font, 1, color, 1)

print("Saturation -- ",numcar)

UpdateLane(fid,flow+numcar,numcar)

else:

loop = 0

except KeyboardInterrupt:

exitFlag = 1

cnt\_up = 0

cnt\_down = 0

pre = 1

url = 'http://192.168.1.8:8080/video'

cap = cv2.VideoCapture('l1.mp4')

# cap = cv2.VideoCapture(url)

# Get width and height of video

w = cap.get(3)

h = cap.get(4)

frameArea = h \* w

areaTH = frameArea / 400

# Lines

line\_up = int(3.1 \* (h / 5))

line\_down = int(3.5 \* (h / 5))

up\_limit = int(1 \* (h / 5))

down\_limit = int(4.3 \* (h / 5))

line\_down\_color = (0xff, 0, 0)

line\_up\_color = (0xff, 0, 0xff)

pt1 = [0, line\_down]

pt2 = [w, line\_down]

pts\_L1 = np.array([pt1, pt2], np.int32)

pts\_L1 = pts\_L1.reshape((-1, 1, 2))

pt3 = [0, line\_up]

pt4 = [w, line\_up]

pts\_L2 = np.array([pt3, pt4], np.int32)

pts\_L2 = pts\_L2.reshape((-1, 1, 2))

pt5 = [0, up\_limit]

pt6 = [w, up\_limit]

pts\_L3 = np.array([pt5, pt6], np.int32)

pts\_L3 = pts\_L3.reshape((-1, 1, 2))

pt7 = [0, down\_limit]

pt8 = [w, down\_limit]

pts\_L4 = np.array([pt7, pt8], np.int32)

pts\_L4 = pts\_L4.reshape((-1, 1, 2))

# Background Subtractor

fgbg = cv2.createBackgroundSubtractorMOG2(detectShadows=True)

c = 0

# Kernals

kernalOp = np.ones((3, 3), np.uint8)

kernalOp2 = np.ones((5, 5), np.uint8)

kernalCl = np.ones((11, 11), np.uint)

font = cv2.FONT\_HERSHEY\_SIMPLEX

cars = []

max\_p\_age = 5

pid = 1

while cap.isOpened():

(ret, frame) = cap.read()

for i in cars:

i.age\_one()

fgmask = fgbg.apply(frame)

fgmask2 = fgbg.apply(frame)

if ret == True:

# Binarization

(ret, imBin) = cv2.threshold(fgmask, 200, 0xff,

cv2.THRESH\_BINARY)

(ret, imBin2) = cv2.threshold(fgmask2, 200, 0xff,

cv2.THRESH\_BINARY)

# OPening i.e First Erode the dilate

mask = cv2.morphologyEx(imBin, cv2.MORPH\_OPEN, kernalOp)

mask2 = cv2.morphologyEx(imBin2, cv2.MORPH\_CLOSE, kernalOp)

# Closing i.e First Dilate then Erode

mask = cv2.morphologyEx(mask, cv2.MORPH\_CLOSE, kernalCl)

mask2 = cv2.morphologyEx(mask2, cv2.MORPH\_CLOSE, kernalCl)

# Find Contours

(countours0, hierarchy) = cv2.findContours(mask,cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE)

for cnt in countours0:

area = cv2.contourArea(cnt)

# print (area)

if area > areaTH:

# ###Tracking######

m = cv2.moments(cnt)

cx = int(m['m10'] / m['m00'])

cy = int(m['m01'] / m['m00'])

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

new = True

if cy in range(up\_limit, down\_limit):

for i in cars:

if abs(x - i.getX()) <= w and abs(y - i.getY()) \

<= h:

new = False

i.updateCoords(cx, cy)

if i.going\_UP(line\_down, line\_up) == True:

cnt\_up += 1

elif i.going\_DOWN(line\_down, line\_up) \

== True:

cnt\_down += 1

break

if i.getState() == '1':

if i.getDir() == 'down' and i.getY() \

> down\_limit:

i.setDone()

elif i.getDir() == 'up' and i.getY() \

< up\_limit:

i.setDone()

if i.timedOut():

index = cars.index(i)

cars.pop(index)

del i

if new == True: # If nothing is detected,create new

p = vehicles.Car(pid, cx, cy, max\_p\_age)

cars.append(p)

pid += 1

cv2.circle(frame, (cx, cy), 5, (0, 0, 0xff), -1)

img = cv2.rectangle(frame, (x, y), (x + w, y + h), (0,

0xff, 0), 2)

for i in cars:

cv2.putText(

frame,

str(i.getId()),

(i.getX(), i.getY()),

font,

0.3,

i.getRGB(),

1,

cv2.LINE\_AA,

)

str\_up = 'UP: ' + str(cnt\_up)

str\_down = 'DOWN: ' + str(cnt\_down)

frame = cv2.polylines(frame, [pts\_L1], False, line\_down\_color,

thickness=2)

frame = cv2.polylines(frame, [pts\_L2], False, line\_up\_color,

thickness=2)

frame = cv2.polylines(frame, [pts\_L3], False, (0xff, 0xff,

0xff), thickness=1)

frame = cv2.polylines(frame, [pts\_L4], False, (0xff, 0xff,

0xff), thickness=1)

cv2.putText(

frame,

str\_up,

(10, 40),

font,

0.5,

(0xff, 0xff, 0xff),

2,

cv2.LINE\_AA,

)

cv2.putText(

frame,

str\_up,

(10, 40),

font,

0.5,

(0, 0, 0xff),

1,

cv2.LINE\_AA,

)

cv2.putText(

frame,

str\_down,

(10, 90),

font,

0.5,

(0xff, 0xff, 0xff),

2,

cv2.LINE\_AA,

)

cv2.putText(

frame,

str\_down,

(10, 90),

font,

0.5,

(0xff, 0, 0),

1,

cv2.LINE\_AA,

)

cv2.imshow('Frame', frame)

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

break

try:

connection = mysql.connector.connect(host='localhost',database='traffic',user='minkhant',password='root')

if connection.is\_connected():

cursor = connection.cursor()

cursor.execute("select database();")

recoord = cursor.fetchone()

query = """SELECT \* FROM lane1 ORDER BY id ASC"""

cursor=connection.cursor()

result = cursor.execute(query)

result = cursor.fetchall()

lanecount = len(result)

if lanecount > rowcount:

for row in result:

delay = row[1]

first\_id = row[0]

now = time.time()

sec = int(now)

key = sec + delay - 3

rowcount = lanecount

print("Key time ", key)

loop = 0

except Error as e:

print("Error While connecting to MySQL")

finally:

if(connection.is\_connected()):

cursor.close()

connection.close()

if key == int(time.time()) and c == 0:

c = 10

flow = cnt\_down - pre

cv2.imwrite(filename='car.jpg', img=frame)

img\_new = cv2.imread('car.jpg', cv2.IMREAD\_GRAYSCALE)

img\_ = cv2.imread('car.jpg', cv2.IMREAD\_ANYCOLOR)

image = cv2.imread('car.jpg')

CountJam(flow,first\_id,loop)

pre = cnt\_down

else:

c = 0

else:

break

cap.release()

cv2.destroyAllWindows()

Above code are available at https://github.com/WannaMinKhant/ThesisPythonCode for two lane phase program.