The project involves developing an autonomous driving controller for an electric vehicle. The driving control model was trained using the model.h5 file. The aim is to use this model to control the vehicle's steering and acceleration.

The old code included an implementation of the SimplePIController class which is responsible for controlling the vehicle's steering. However, the code contained some errors and needed improvements to function correctly.

The following modifications were made to the old code:

I removed the import of the h5py library as it was not used in the code.

I updated the fixed speed to 20.

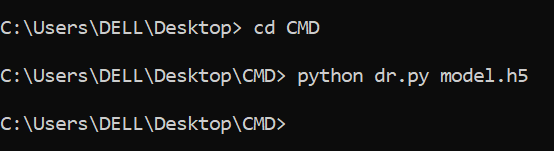
I added prints to display the steering and acceleration values after they have been updated.

I updated the code to save the image only if the image directory is not empty.

I adjusted the SimplePIController class to set the target value to 20.

I defined the values for the variables vehicle\_mass, brake\_deadband, decel\_limit, accel\_limit, wheel\_radius, wheel\_base, steer\_ratio, max\_lat\_accel, and max\_steer\_angle.

But the new code have a problem because when i run it in the shell he don’t told me any thing, he just goes back to the line like that :



So I hope that I can fix this problem the next session.

The new code :

import argparse

import base64

import json

import numpy as np

import socketio

import eventlet

import eventlet.wsgi

import time

from PIL import Image

from PIL import ImageOps

from flask import Flask, render\_template

from io import BytesIO

from keras.models import load\_model

import h5py

# Fix error with Keras and TensorFlow

import tensorflow as tf

tf.compat.v1.disable\_eager\_execution()

sio = socketio.Server()

app = Flask(\_\_name\_\_)

model = None

prev\_image\_array = None

class VehicleParams:

    def \_\_init\_\_(self):

        self.vehicle\_mass = 2500 # kg

        self.brake\_deadband = .1 # range [0,1]

        self.decel\_limit = -5 # m/s^2

        self.accel\_limit = 1 # m/s^2

        self.wheel\_radius = 0.3 # meters

        self.wheel\_base = 2.8 # meters

        self.steer\_ratio = 14.8

        self.max\_lat\_accel = 3 # m/s^2

        self.max\_steer\_angle = 8 # degrees

class SimplePIController:

    def \_\_init\_\_(self, Kp, Ki):

        self.Kp = Kp

        self.Ki = Ki

        self.set\_point = 20

        self.error = 0.0

        self.integral = 0.0

    def run(self, measurement):

        self.error = self.set\_point - measurement

        self.integral += self.error

        return self.Kp \* self.error + self.Ki \* self.integral

class Controller:

    def \_\_init\_\_(self, vehicle\_params):

        self.vehicle\_params = vehicle\_params

        self.steering\_controller = SimplePIController(0.1, 0.002)

        self.throttle\_controller = SimplePIController(0.3, 0.1)

    def control(self, current\_velocity, dbw\_enabled, linear\_velocity, angular\_velocity):

        if not dbw\_enabled:

            self.throttle\_controller.reset()

            return 0., 0., 0.

        current\_velocity = self.velocity\_mps\_to\_kmph(current\_velocity)

        steering = self.steering\_controller.run(current\_velocity)

        throttle = self.throttle\_controller.run(linear\_velocity)

        brake = 0

        if linear\_velocity == 0. and current\_velocity < 0.1:

            throttle = 0

            brake = 700 # N\*m - to hold the car in place if we are stopped at a light. Acceleration - 1m/s^2

        elif throttle < .1 and abs(current\_velocity - linear\_velocity) < .1:

            throttle = 0

            decel = max(self.vehicle\_params.decel\_limit,

                        self.vehicle\_params.accel\_limit)

            brake = abs(decel) \* self.vehicle\_params.vehicle\_mass \* self.vehicle\_params.wheel\_radius # Torque N\*m

        return steering, throttle, brake

    def reset(self):

        self.steering\_controller.reset()

        self.throttle\_controller.reset()

    def velocity\_mps\_to\_kmph(self, velocity\_mps):

        return velocity\_mps \* 3