**Project 3: Location based perception and action in Guardian Angel for Autonomous Driving**

This project has two tasks:

**Task 1:** Integrate google services with your Project 1 app to monitor traffic conditions along the way for a given starting point and destination and **continuously assess** whether the driving condition is normal or poor and consequently did it induce a low cognitive or high cognitive workload.

You will create a new Activity in the app, that takes a start address and end address.

It then calls the Google Map API to obtain a path from start to end address.

Along the path, with the default sample rate get the traffic congestion data.

Design a **clustering algorithm** that categorizes the traffic congestion data into normal and poor road condition.

Consequently, obtain a categorization of low cognitive and high cognitive workload.

**Task 2:** Consider a Vehicular Ad Hoc Network (VANET) where three vehicles are connected to each other as shown in Figure 1.

A red car in a parking lot

Description automatically generated

In this case, the car Z is disabled. Car A starts decelerating following some algorithm. This algorithm is unknown to Car B and also unknown to you as a designer of the autonomous braking system. Car A’s braking model will be given to you without access to its actual control design. You have to design a Fuzzy logic based advisory control for autonomous braking in Car B.

**Advisory Control System Model**

**Input:** Velocity from CAR A.

**Logic of advisory control**

Aim: Compute its own deceleration based on three criteria

*Criteria 1:* Estimate of the braking force of CAR A, whether it is doing heavy or medium or low braking

*Criteria 2:* Estimate of the distance between CARs A and B, whether it is far or close

*Criteria 3: Estimate of the road condition based on Task 1 of the project.*

These three criteria can be combined in a fuzzy inference system to determine the deceleration that the CAR B should have.

The next aim is to achieve this deceleration through the advisory control strategy designed in Project 2.

We can use a switching rule based on the deceleration that the CAR B should have to avoid collisions.

In this project we will use the following rule

If CAR B deceleration from fuzzy inference system > 0.75 deceleration limit

Then switch to the human, where the human provides 1.1 deceleration limit

Else

Remain with the autonomous vehicle

**Output:**

1. Switch to human or remain with autonomous vehicle.
2. If switch to human then what time the switch occurs

**Deployment logic of Advisory Control**

Assume that the Car B starts with autonomous mode.

The scenario consists of: **A)** road condition: Normal or Poor and **B)** Two initial speeds: speed of CAR A, and speed of CAR B.

You run the scenario by running the Simulink model “LaneManageSystem3Car.slx” after configuring for road condition and initial speeds of both the cars.

For a scenario, the advisory control runs every time sample. The time samples will be given by the Simulink model.

For each time sample, the advisory control will decide if it should switch to the human.

If the advisory control decides not to switch, you continue the autonomous controller execution through the Simulink model.

If the advisory control decides to switch, then you have to obtain two inputs:

1. Time at which the advisory control decides to switch
2. Reaction time of the human user 3

From time , to the time of switch you use the autonomous control to brake

After the time of switch, you abort the autonomous control and instead use the human to brake for the rest of the simulation. This is done by the Simulink model “Level3Model.slx”.

**Fuzzy Logic Helper Functions**

You will implement fuzzy logic controller using the fuzzy logic toolbox. To help with your implementation here are functions that give you membership values.

Function membershipFunctionBrakingNA

It takes the deceleration value as input (make sure to only pass a positive value) and provides a vector with membership values of low, medium and hard braking states, respectively.

You can use this function for both car A and B.

Function membershipFunctionDistance

It takes the distance between Car A and B as input (make sure to pass a negative value) and provides a vector with membership of far and close, respectively

Function membershipFunctionRoad

It takes the speed difference between average speed and current speed as obtained from the Google API call in Task 1 and gives membership of poor and normal road, respectively

**Simulink Models**

Model 1: LaneManageSystem3Car.slx

It simulates the 3 car scenario. You have to configure the initial speeds of CAR A and B using the following instructions:

set\_param('LaneMaintainSystem3Car/VehicleKinematics/Saturation','LowerLimit',num2str(decelLim))

set\_param('LaneMaintainSystem3Car/VehicleKinematics/vx','InitialCondition',num2str(initSpeedB))

set\_param('LaneMaintainSystem3Car/CARA/VehicleKinematics/Saturation','LowerLimit',num2str(decelLim))

set\_param('LaneMaintainSystem3Car/CARA/VehicleKinematics/vx','InitialCondition',num2str(initSpeedA))

You should not change any other parameters of this Simulink model

Model 2: Level3Model.slx

It simulates a scenario, where the autonomous control is in charge for time and then the advisory control switches to human control, which remains inactive for an additional time and then provides deceleration at the rate of

You should setup the Level3Model.slx using the following instructions.

set\_param('Level3Model/VehicleKinematics/Saturation','LowerLimit',num2str(100\*decelLim))

set\_param('Level3Model/VehicleKinematics/vx','InitialCondition',num2str(initSpeedB))

set\_param('Level3Model/CARA/VehicleKinematics/Saturation','LowerLimit',num2str(decelLim))

set\_param('Level3Model/CARA/VehicleKinematics/vx','InitialCondition',num2str(initSpeedA))

set\_param('Level3Model/Constant1','Value',num2str(timeToSwitchToHuman))

set\_param('Level3Model/Step','Time',num2str(HumanReactionTime+timeToSwitchToHuman))

set\_param('Level3Model/Step','After',num2str(1.1\*decelLim))

**Testing Logic**

We will only test for two scenarios, Low cognitive workload (LCW) and high cognitive workload (HCW) separately. Just like Task 1 od Project 2, you have to test for a large number of scenarios. The scenario as explained before has two factors: a) normal and poor road conditions which directly correspond to LCW and HCW, and b) two initial speeds: one for CAR A and another for CAR B, each in the range of [20 60].

**Deliverables**

A matlab file named AdvisoryControl.m that is a function of the following form

Function [switch, timeOfSwitch] = AdvisoryControl(,)

This function takes a vector of velocities of A and a vector of the distances between A and B for a time interval at sampling frequency (this can be variable) and decides whether it wants to switch to human and also provides the time when the Advisory control wants to switch.