Modeling Semi-Autonomous Vehicle Driving for Indian Terrain

Avinash Chambhare, *CSE,IIT Kanpur,* Satyam Dwivedi, *EE,IIT Kanpur,*
and Utsav Gupta, *EE,IIT Kanpur,*
Indranil Saha, *Professor,IIT Kanpur*

***Abstract*—Modern passengers vehicles are quipped with computational, sensing and actuating capabilities which enable the design of systems that can autonomously control the vehicle motion in order to assist the driver. In this project we intended to develop a safety system which utilizes a predictive controller to guide a vehicle through an obstacle free path in a safe manner. This is a combined lane keeping, stability control and collision avoidance problem.**

***Keywords*—Extended Kalman Filter, A* algorithm, PID controller, Obstacle modelling.**

I. Introduction

Modern passenger vehicles are equipped with computational, sensing and actuating capabilities which enable the design of systems that can autonomously control the vehicle motion in order to assist the driver. In this project we intended to develop a safety system which utilizes a predictive controller to guide a vehicle through an obstacle free path in a safe manner. This is a combined lane keeping, stability control and collision avoidance problem. An important part of developing such a system is threat assessment. Threat assessment refers to evaluating in real time if and when an accident is unavoidable. The model should take into account the limitations specified in the vehicle model to identify such conditions. We also wish to develop a state to which the model will the transition before giving control of the vehicle to the driver on encountering a threat condition. The vehicle model that we use results in a non-linear system. However we transform it into a linear system and this allows us to use the Proportional-Integral-Derivative (PID) Controller for control design. We chose to develop our model for an Indian traffic environment because such an environment provides greatly varying and dynamic obstacles. Such an environment provides a perfect template to design a comprehensive traffic model. Performance of the system in such traffic models will help us test its robustness.

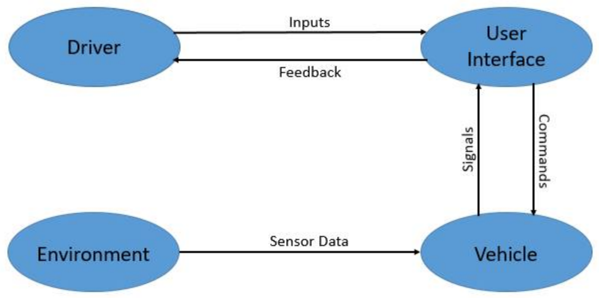
II. Problem Description

The problem is divided into two parts: i) Designing conceptual behaviour of the vehicle, ii) Designing physical model that could accomplish above behavior. In the first part we would like to decide the overall state flow of our vehicle, that is, how it is going to behave under different environmental conditions. It would get observations and decide objectives to be passed to the physical level to execute. The physical level is modelled to complete the following task:

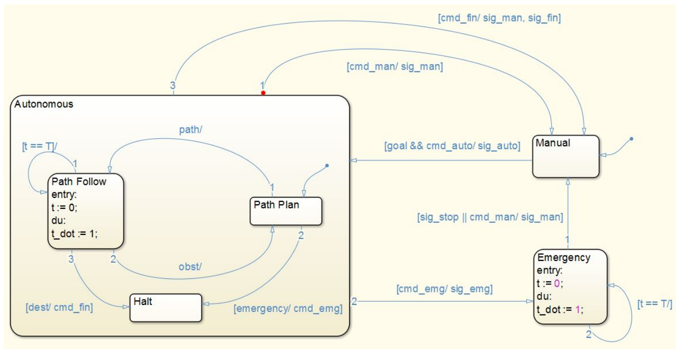
We are given a 2D map of the environment along with the start point and end point. The start point is the point in the map from which the vehicle begins its journey with the end point being the destination. The sensors in the vehicle provide us with information about the environment which help us detect obstacles in the road and localize our position in the map. Our objective is to guide the vehicle through the obstacles in a safe way (i.e follow the safety constraints at all times) from the start point to the end point by controlling the velocity and the steering angle of the vehicle or take appropriate action if anything unexpected happens.

III. Proposed Solution

Before proceeding to the solution we have developed for the problem described, we need to define the different agents that constitute the solution space and understand their interaction with each other.



We define *Driver, Vehicle* and the *Environment* as the discreete agents and their interactions have been shown above. The driver can communicate with the vehicle through a user interface by giving it inputs which are translated as commands by the interface and sent to the vehicle. The actions taken by the vehicle on these commands generate signals which the interface recieves and provides appropriate feedback to the driver. The environment is another agent that interacs with the vehicle. The sensors in the vehicle scan the environment and the data recieved is recorded and processed. We now give the state flow of the proposed solution:



The model overall has three main states: *Autonomous, Manual* and the *Emergency* states. The *Manual* state represents the case where the driver of the vehicle has its control. In the *Autonomous* state, an autonomous system is responsible for the control of the car. The *Emergency* state is an intermediary state reached when the control of the car is to be returned back to the driver by the autonomous sytem in case of an emergency it is unable to handle.

We assume that the system initially is in the *Manual* state where the driver is driving the car. It can go to the *Autonomous* state when the driver issues a `cmd_auto` command. The driver is also required to provide the co-ordinates of the destination which are encoded in the input `goal`. The vehicle then generates a `sig_auto` signal to notify the driver that the system is in the *Autonomous* state. Inside the *Autonomous* state, the initial state is *Path Plan.* In this state, the path to be followed by the vehicle is generated. This path is the input to the *Path Follow* state where the vehicle follows this path for time T intervals where,

$$T = \frac{1}{f},$$

f is the sampling frequency of the system, until an obstacle is encountered. So, on input `obst`, transition to *Path Plan* takes place and an updated path is generated.

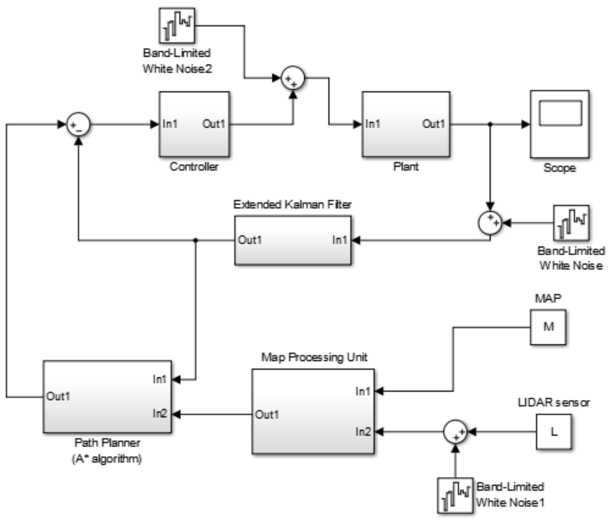
A transition to the *Halt* state takes place when the destination is reached in the *Path Follow* state or when the no feasible plan can be generated in the *Path Plan* state. On successfully reaching the destination, the system *Halts* and a `cmd_fin` command is generated. On getting into an emergency (no feasible path is found), system *Halts* generating a `cmd_emg` command. The *Autonomous* state on `cmd_fin` transitions back to the *Manual* state and the `sig_man` and `sig_fin` signals are generated. On `cmd_emg`, transition to the *Emergency* state takes place and the `sig_emg` signal is generated.

The *Emergency* state is reached on `cmd_emg` and the system stays in this state until the `sig_stop` signal (signalling system stop) or `cmd_man` command is recieved. The control of the vehicle is then handed back to the driver with a transition to the *Manual* state.

We have mentioned that the driver can seize control of the system when it is in the *Autonomous* state. This is enabled by the pre-emptive transition from the *Autonomous* state to the *Manual* state on the `cmd_man` command. Thus the given state flow captures all the behavioral aspects of the proposed model. In the next section,modelling of the autonomous part of the system is described.

IV. Autonomous System Modelling

As stated earlier, we use a PID controller to control the velocity and steering angle of the vehicle to safely guide it through the obstacles. The overall block diagram of the system is shown below:



Each of the Individual blocks shown above are described below: The vehicle is our plant. The state of the vehicle in 2D space is given by (x, y, θ). Here (x, y) is the position of the vehicle in the x, y plane and θ is the angle of its orientation. We denote the velocity of the vehicle as v and the steering angle as δ. $u = (v, \delta)$ is the control variable. The equations of motion of the vehicle are given by:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} vcos(\alpha(\delta)+\theta) \\ vsin(\delta)+\theta) \\ a/btan\delta \end{bmatrix}$$

$$\alpha(\delta) = arc(tan(\frac{a \times tan(\delta)}{b}))$$

where v_o is the velocity of the rear wheel, b is distance between rear and front wheels and $\alpha(\delta)$ is the angle between velocity vector and the vehicle length axis.

A. Noise Generator

The position of the vehicle is determined using the data received by GPS and IMU sensors. The data from these sensors is susceptible to contamination by noise. So, in order to simulate the data from these sensors, we use a Noise Generator to add noise to the data about the state of the vehicle. The noise signal added is a White Gaussian Noise (WGN).

B. Extended Kalman Filter (EKF)

The observed vehicle state data plus the noise from the Noise Generator are provided to the EKF block. In this block we apply the Extended Kalman Filter algorithm on the noisy input to generate a good estimate of the actual robot state. EKF works in two steps: prediction and update. In the first step it uses its previous state, the control action taken and the vehicle model to predicts its current state. In the second step it takes the sensor information and fuse it with the prediction information to obtain fairly accurate estimate of the actual state. EKF is widely used in mobile robot localization.

C. Map Processing Unit

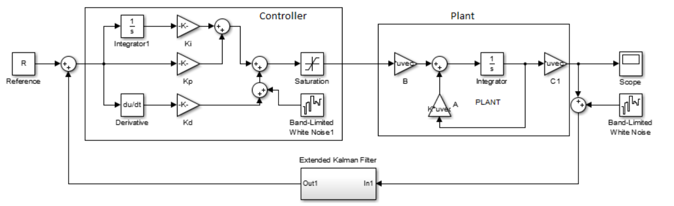
The Map Processing Unit (MPU) has two inputs. One input to the MPU is the overall 2D map of the environment with the start point and the end point specified. The other input is the data from the LIDAR sensor. The MPU uses this data from LIDAR to identify the obstacles and combines it with the map which is then fed forward to the Path Planner block.

D. Path Planner

The Path planner receives from the MPU the map of the obstacles. It uses the A* path planning algorithm to generate a path for the vehicle to follow for safe traversal. The path planner also takes care of the follow safety constraint: Keep the vehicle on a collision free path within the lane. To keep the vehicle from colliding with the obstacles, we maintain a certain minimum clearance distance between them. This is done by introducing padding around the obstacles. The A* then provides us with a path that avoids the obstacles. We take the Euclidean distance between two points to be the heuristic for the path planning algorithm.

E. Controller

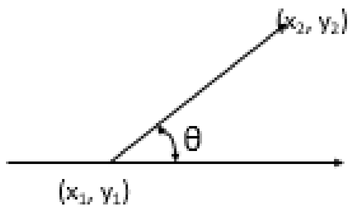
As stated earlier, the velocity of the vehicle v, and the steering angle are the control variables. The following figure shows the Controller in detail



We have used a PID controller for our control design. The equations that describe the vehicle model constitute a non-linear system. Although controller that handle system non-linearity exist, it is much easier to handle linear systems. So, we remove the non-linearity in the system by using a Transform Matrix. By doing this, the aim of the controller becomes to move the bot along the x-axis, a comparatively easier task.

F. Details of Implementation

The EKF, Path Planner and Controller blocks were implemented in MATLAB. The Transform Matrix that linearizes the system is given below



$$T = \begin{bmatrix} cos\theta & -sin\theta & 0 & x_1 \\ sin\theta & cos\theta & 0 & y_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \ X = \begin{bmatrix} x \\ y \\ \theta \\ 1 \end{bmatrix}$$

$$\theta = tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right), \ X' = T^{-1}X$$

X' is the representation of the point X in the transformed axis which is rotated at an angle θ. The transfer function $G(s)$ is calculated and comes out to be

$$G(s) = C(sI - A)^{-1}B,$$

$$G(s) = \alpha \times \frac{b}{as}, \ where$$

$a = 4, \ b = 2, \ \alpha = 2$ We discretized the controller using first order hold and a sampling period of $T = 0.025$ secs The values of the coefficients for the proportional, integral and derivative terms are given respectively by

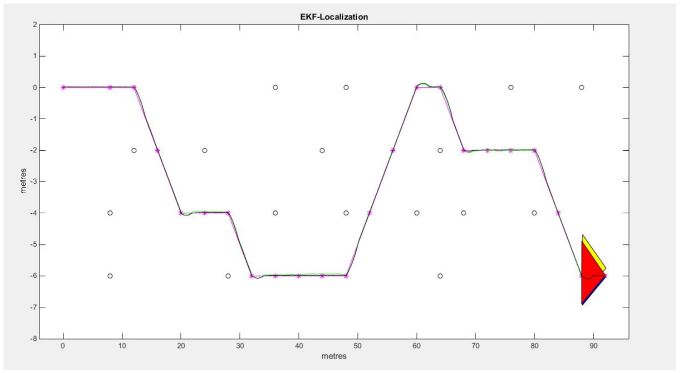
$$K_p = 35.5, \ K_i = 0.056, \ K_d = 0.02$$

G. Experimental Results

Major part of the project was to develop an autonomous robot model that could also respond to the signals from the environment and function according to the state diagram given in section 2. To simulate this part, we have created a model of the environment and the vehicle itself, as explained in section 3, in MATLAB and performed simulations by giving random obstacle positions. We observed that our simulated vehicle was able to reach its pre-decided goal while avoiding all the obstacles in minimum time possible, by taking the shortest path. The simulation video is uploaded here:

https://youtu.be/E2d0HV8RY6s

Below is the snapshot of the simulation environment after the robot reaches its destination.



V. Lessons Learnt

Many of the concepts taught in the class proved helpful while working on this project. The *State Model of Computation* taught in class was instrumental in helping us formulate and capture the aspects that we desired of the model we designed. The plant vehicle model that we designed was based on a similar vehicle model that we encountered in one of the assignments of the course. The design of the controller was also aided by the class teachings. Also, we were introduced to *MATLAB* and *Stateflow* in the first couple of classes. These were essential tools for us while working on this project.

VI. Future Scope

Currently, the system that we have implemented has a very simplistic traffic model where the obstacles are stationary. We can extend our project to develop a more accurate traffic model that incorporates mobile obstacles and other conditions such as traffic light stops. Try to develop driver models to account for attentive and distracted driving and then enhance the capabilities of the autonomous mode that enables it to take corrective action in case of distracted driving and/or suggest optimal paths along which to drive when the driver is attentive. To come up with appropriate actions that the system can take in case it is not possible to follow the safety constraints. Implement the model on a miniature scale to test its feasibility for real life application.

Acknowledgment

We would like to thank Professor Indranil Saha, all the members of our team and TA of the course CS637: Embedded and Cyber Physical system, who helped us during the whole project and provided us regular and crucial feedback in the process. It would be impossible to complete this project without their valuable support.

References

[1] Semi-Autonomous Vehicle Control for Road Departure and Obstacle Avoidance Andrew Gray, Mohammad Ali, Yiq Gao, J. Karl Hedrick, Francesco Borrelli, University of California, Berkeley, USA

[2] Seshia, Sanjit A., Dorsa Sadigh, and S. Shankar Sastry. ”Formal methods for semi-autonomous driving.” Proceedings of the 52nd Annual Design Automation Conference. ACM, 2015.

[3] Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. ”A formal basis for the heuristic determination of minimum cost paths.” IEEE transactions on Systems Science and Cybernetics 4.2 (1968): 100-107.

[4] Sadigh, Dorsa, et al. ”User interface design and verification for semi-autonomous driving.” Proceedings of the 3rd international conference on High confidence networked systems. ACM, 2014.

[5] A* algorithm demo provided by matlab here: https://in.mathworks.com/matlabcentral/fileexchange/26248-a—a-star–search-for-path-planning-tutorial