Autonomous Vehicles

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Production and Industrial

by

Anmol Agarwal

17BPI0081

Under the guidance of

Prof. C Ramesh Kumar

ARC VIT, Vellore.

June, 2021

DECLARATION

I hereby declare that the thesis entitled "Autonomous Vehicle" submitted by me, for the award of the degree of *Bachelor of Technology in Production and Industrial* to VIT is a record of bonafide work carried out by me under the supervision of Prof C Ramesh Kumar.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date :3-06-21

ANMOL AGARWAL

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled "Autonomous Vehicle" submitted by **Anmol Agarwal & 17BPI0081**, **SMEC**, VIT University, for the award of the degree of *Bachelor of Technology in Production and Industrial*, is a record of bonafide work carried out by him under my supervision during the period, 01. 02. 2021 to 30.05.2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date :3-06-21

C. RAMESH KUMAR

Signature of the Guide

Internal Examiner

External Examiner

Head of the Department

Programme

ACKNOWLEDGEMENTS

I would like to thank Prof. C Ramesh Kumar for his guidance and brilliant insights in the project, for which I am eternally grateful.

I would like to thank Professors who conducted our review and provided us with valuable feedback which helped us with improving the quality of the work.

I would like to thank my project partner for continued hard work.

Anmol Agarwal

Executive Summary

This document will be regarding the design and development of longitudinal and lateral vehicle control for autonomous vehicles. Both of these controls will be researched and developed at the V-CARE facility in the VIT Vellore campus.

The work carried out was divided into two aspects

Longitudinal Vehicle control:

- This controller is used in AV for controlling throttling and breaking on the vehicle.
- This functionality can be crystallized by the following methods:
    - Proportional – Integral – derivative (PID) controller
    - Feed forward controller

Lateral Vehicle control:

- This controller in used in AV for controlling steering systems on the vehicle.
- This functionality can be crystalized by the following methods:
    - Geometric controllers
        - Pure pursuit
        - Stanley
    - Dynamic controller
        - Model predictive controller (MPC)
        - Feedback - Feedforward

After that a reference path was chosen inside the VIT campus and a simulation was prepared along with simulation on CARLA simulator and MATLAB simulator.

CONTENTS

Page No.

	Acknowledgement	i
	Executive Summary	ii
	Table of Contents	Iii
1	**INTRODUCTION**	1
	1.1 Objective	7
	1.2 Motivation	8
	1.3 Background	10
2	**PROJECT DESCRIPTION AND GOALS**	25
3	**TECHNICAL SPECIFICATION**	27
4	**DESIGN APPROACH AND DETAILS (as applicable)**	29
	4.1 Design Approach / Materials & Methods	32
	4.2 Codes and Standards	35
5	**PROJECT DEMONSTRATION**	40
6	**COST ANALYSIS / RESULT & DISCUSSION (as applicable)**	**41**
7	**SUMMARY**	**46**
8	**REFERENCS**	**47**

CAPSTONE PROJECT – AUTONOMOUS VEHICLE SYSTEMS
SCOPE – LATERAL AND LONGITUDINAL VEHICLE CONTROL

14-02-2021

ABSTRACT

This document will be regarding the design and development of longitudinal and lateral vehicle control for autonomous vehicles. Both of these controls will be researched and developed at the V-CARE facility in the VIT Vellore campus.

INTRODUCTION

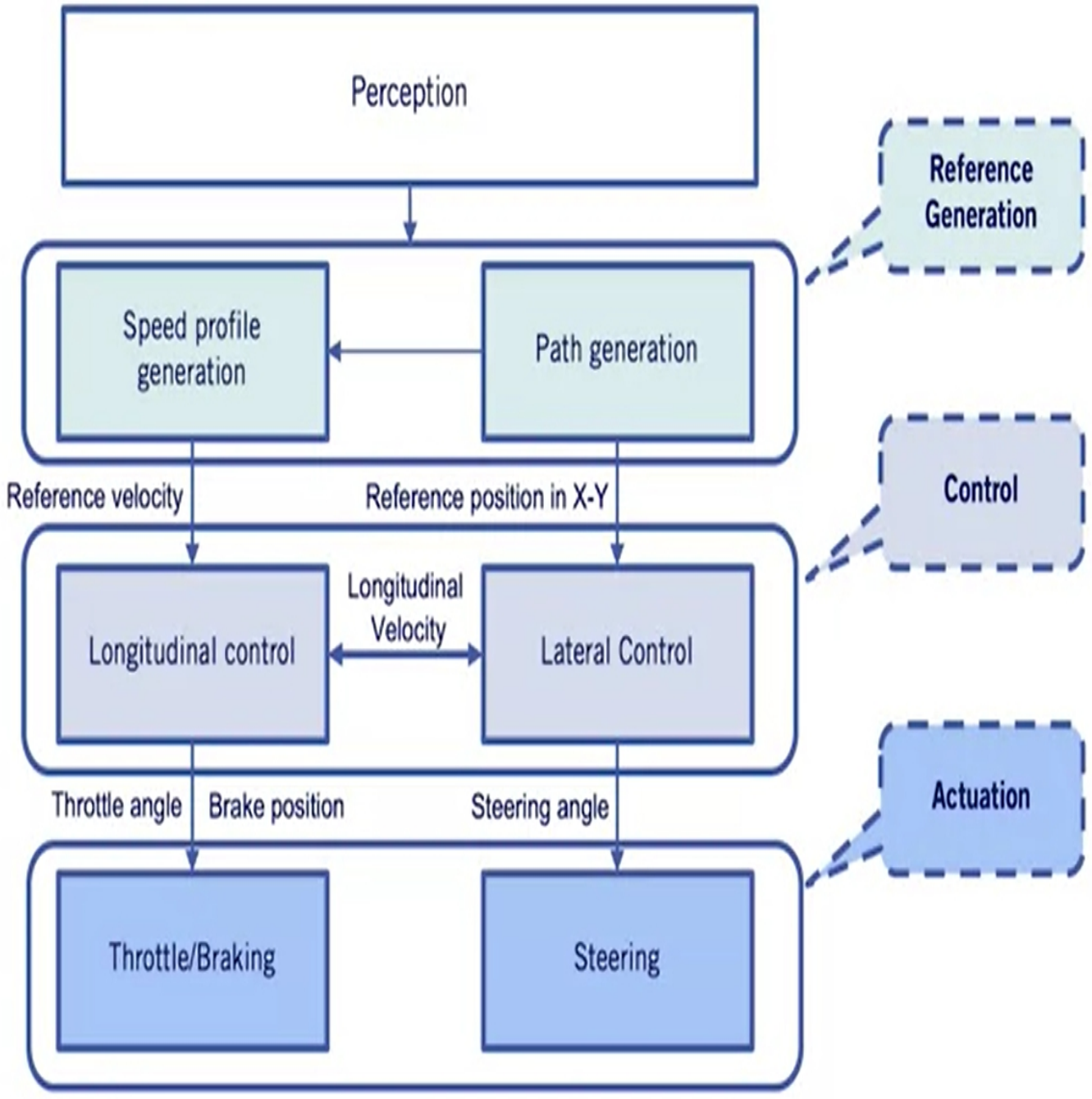
1. Longitudinal Vehicle control:

- This controller is used in AV for controlling throttling and breaking on the vehicle.
- A good example would be Adaptive cruise control for moderating speed of the vehicle on a highway where the vehicle can detect obstacles in the front and back of the vehicle and abiding to the constraints inserted by the driver, decide the vehicle speed and velocity profile.
- This functionality can be crystallized by the following methods:
    - Proportional – Integral – derivative (PID) controller
    - Feed forward controller

2. Lateral Vehicle control:

- This controller in used in AV for controlling steering systems on the vehicle.
- A good example would be Adaptive Lane keeping maneuver where the vehicle can detect the lane patters on the highway and moderate the steering angle to adjust to the same lane as the path progresses, this can also help in changing lane when you want to exit the highway.
- This functionality can be crystalized by the following methods:
    - Geometric controllers
        - Pure pursuit
        - Stanley
    - Dynamic controller
        - Model predictive controller (MPC)
        - Feedback - Feedforward

Architecture of Vehicle Control Strategy



3. Longitudinal Vehicle control:

Proportional – Integral – derivative (PID) controller

Proportional-Integral-Derivative Controller

- In the time domain:

$$u(t) = K_P e(t) + K_I \int_0^t e(t)dt + K_D \dot{e}(t)$$

where K_P, K_I, K_D are the proportional, integral and derivative gains

- In the Laplace domain:

$$U(s) = G_c(s)E(s) = \left(K_P + \frac{K_I}{s} + K_D s\right)E(s)$$
$$= \left(\frac{K_D s^2 + K_P s + K_I}{s}\right)E(s)$$

Proportional-Integral-Derivative Controller

$$G_c(s) = \frac{K_D s^2 + K_P s + K_I}{s}$$

two zeros added

one pole added

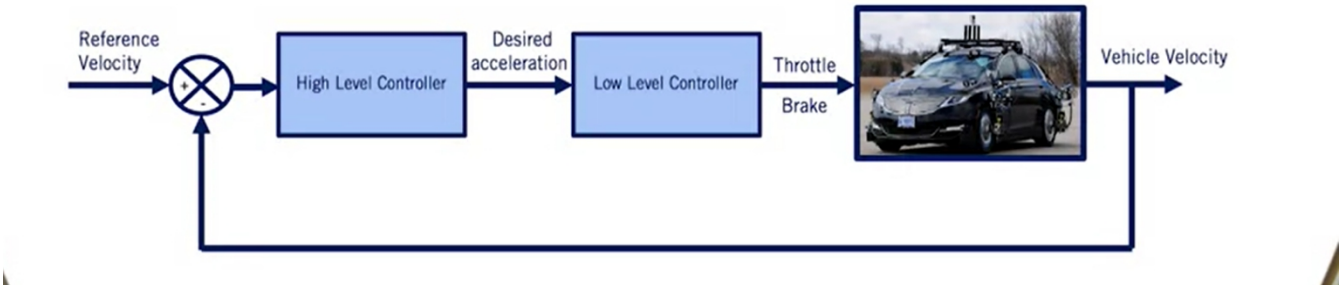
- The pole is at the origin
- The zeros can be arbitrary places on the complex plane
- PID controller design selects zero locations, by selecting P, I, and D gains
- There are several algorithms to select the PID gains (e.g. Zeigler-Nichols)

Characteristics of P, I, and D Gains

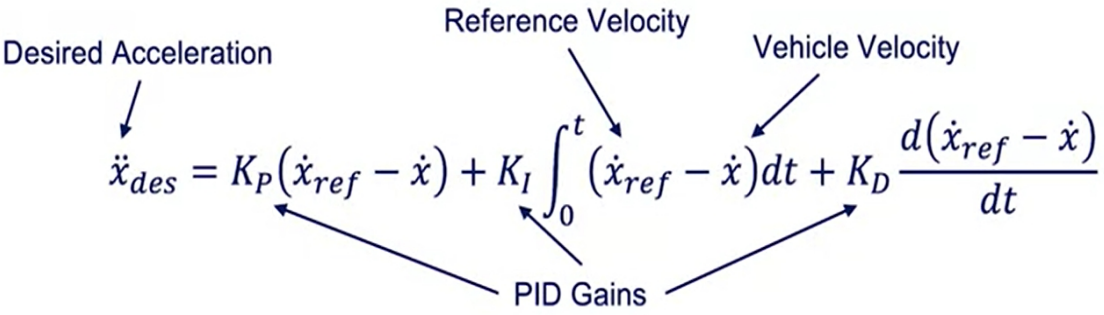
Closed Loop Response	Rise Time	Overshoot	Settling Time	Steady State Error
Increase K_P	Decrease	Increase	Small change	Decrease
Increase K_I	Decrease	Increase	Increase	Eliminate
Increase K_D	Small change	Decrease	Decrease	Small change

- A proportional term directly proportional to the error E, an integral term proportional to the integral of the error, and a derivative term proportional to the derivative of the error. The constants Kp, Ki, and Kd are called the proportional integral and derivative gains and govern the response so the PID controller which is denoted U of t as it is the input to the plant model.
- By adding these three terms of the PID controller together, we get a single transfer function for PID control. Note that not all gains need to be used for all systems. If one or more of the PID gains are set to zero, the controller can be referred to as P, Pd or Pi. The PID transfer function contains a single pole at the origin which comes from the integral term.
- Rise time, overshoot, settling time, steady state error needs to be perfectly tuned according to the state estimation and localization of the AV to work optimally, as discrepancy in any of the such can lead to system overload and/or failure of the longitudinal systems control.

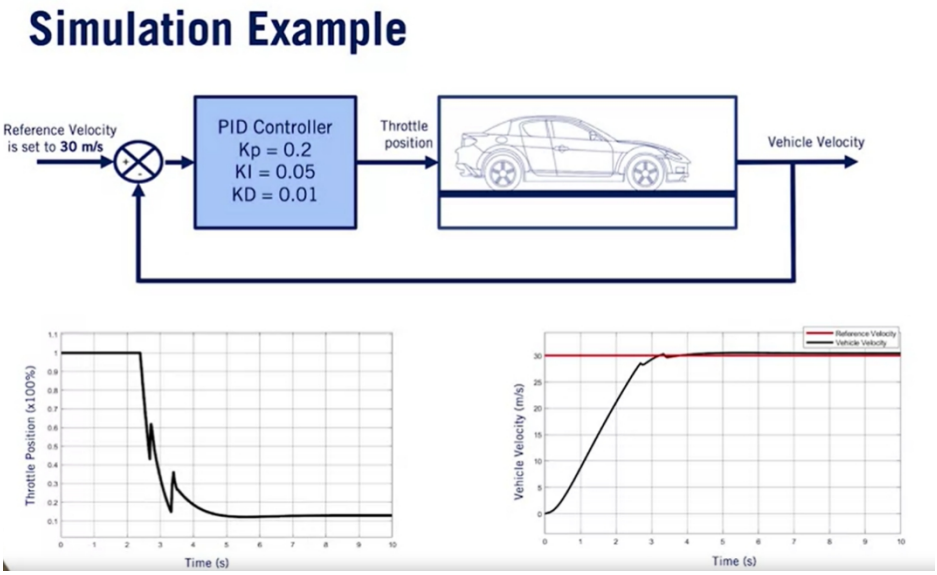
PID Implementation



- The high-level controller determined the acceleration required by the vehicle based on reference and actual velocity.



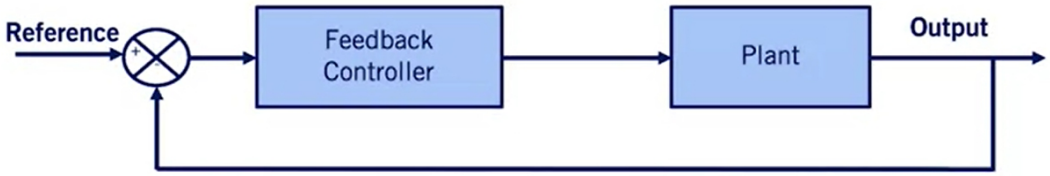
- The low-level controller generates the throttle and breaking signals to follow the desired acceleration calculated by the high-level controller.
- The driver can intervene when and if it required.
- The low-level controller when provided with the acceleration data from the high-level controller calculated the throttle angle in real-time and sends the command to the motor controller to execute the PID controller command.



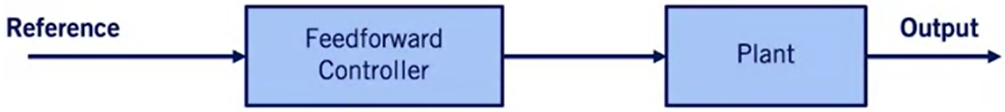
- modify our control architecture to incorporate feedforward commands, which will improve tracking performance, particularly in dynamic manoeuvres.

Feedback vs. Feedforward Control

Feedback - Closed Loop



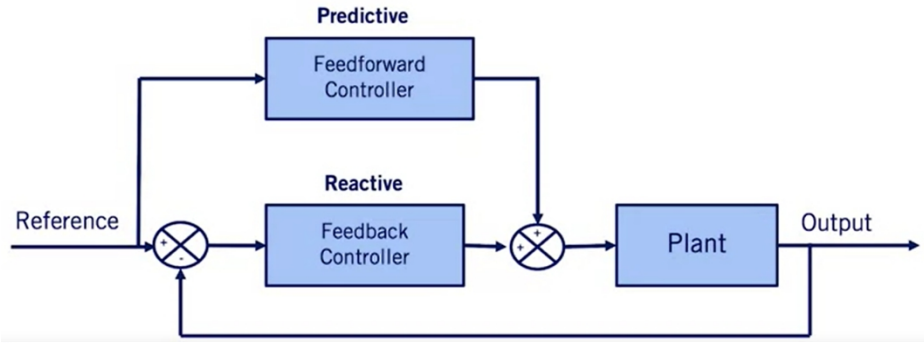
Feedforward - Open Loop



- The feedback block diagram shows the typical closed loop structure, where the current output is compared to a reference signal. And the error between the two is fed into the feedback controller, which generates the input to the plants.
- Feedback controller corrects the response, compensate for disturbances and errors in the model.
- Feed forward controller provides predictive response, non – zero offset.
- The feedforward block diagram shows an open loop structure, where the reference signal is directly fed into the feedforward controller, which again, generates the inputs to the plant.

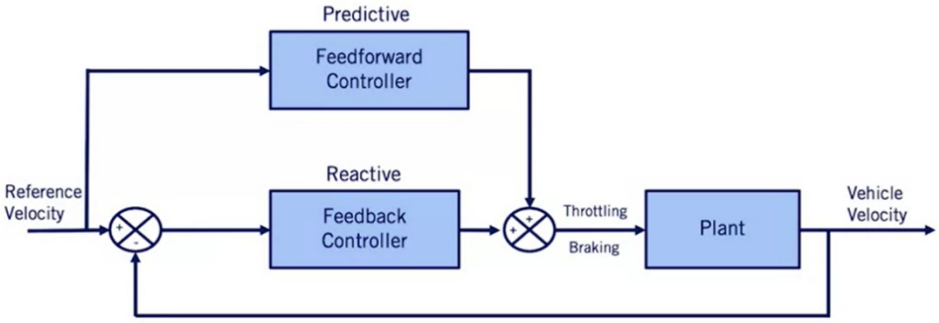
Combined Feedforward and Feedback Control

- Feedforward and feedback are often used together:
  - Feedforward controller provides predictive response, non-zero offset
  - Feedback controller corrects the response, compensating for disturbances and errors in the model



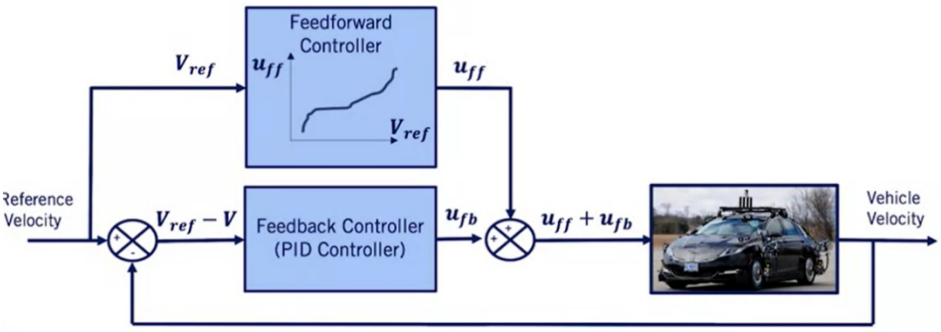
Vehicle Speed Control

- Throttling & Braking:
  - The output of the feedforward and feedback control blocks are the throttling or braking signals to accelerate or decelerate the vehicle (plant) to keep the vehicle velocity close to the reference velocity.

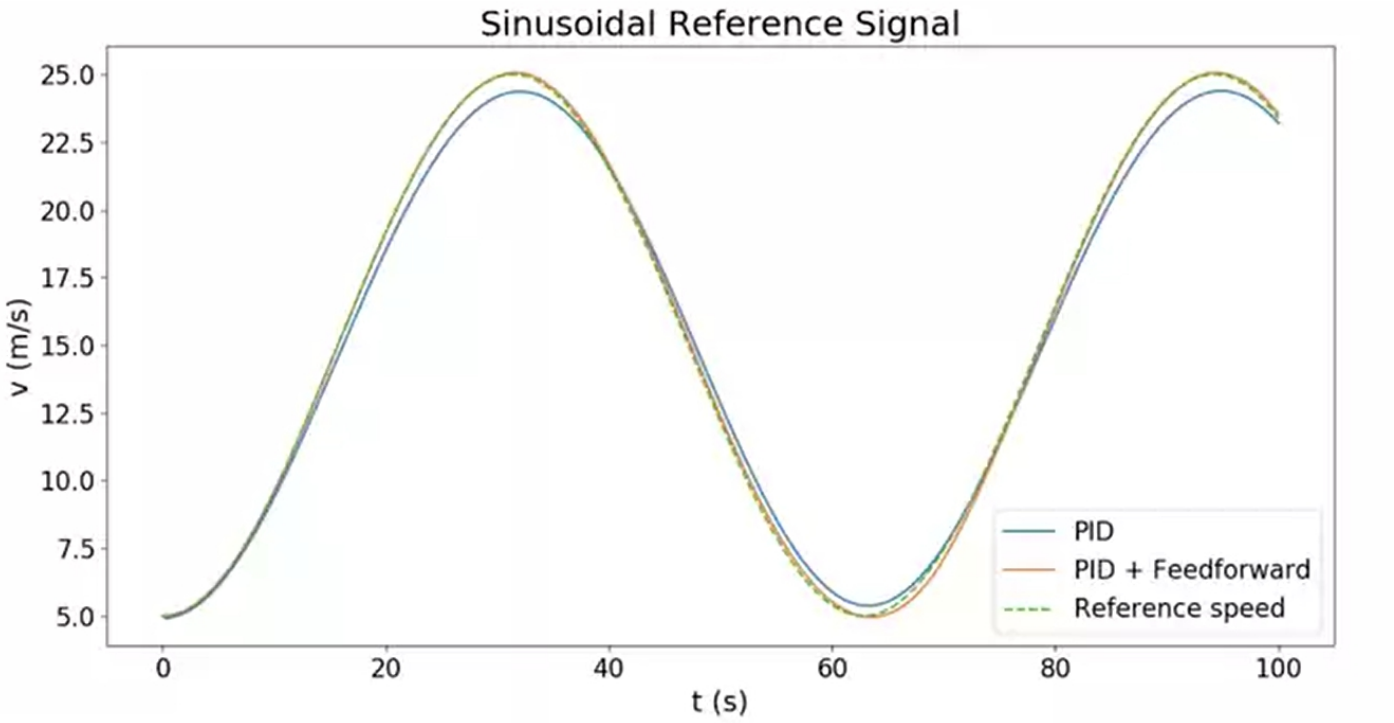


Controller Actuators

- Actuators (throttle angle):
  - The feedforward controller generates the actuator signal (u_{ff}) based on the predefined table and the feedback controller generates the actuator signal (u_{fb}) based on the velocity error.



Feedforward Simulation Results



COMPARISION BETWEEN PID AND PID + FEEDFORWARD CONTROL

The comparison between PID control method and the combined feedforward, feedback method. The key difference between the two responses is visible as the reference speed changes. The PID controller needs errors to exist before it can correct them, its response lags the feedforward approach, which immediately applies the relevant input reference values.

The feedforward tracking is still not perfect, however, as the vehicle response is ultimately governed by its inertia, and the feedforward approach relies on steady state modelling of the car. As the feedforward model becomes more precise, the feedback components can focus purely on disturbance rejection, and speed profile tracking can be done with consistent precision.

4. Lateral Vehicle control:

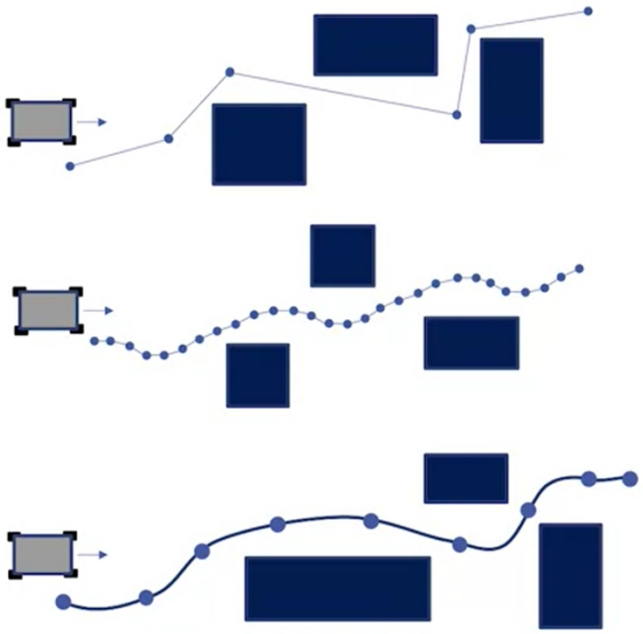
- It defines error relative to desired path
- Select a control law that drives error to zero satisfies input constraints
- Add dynamic considerations to manage forces moments acting on the vehicle

Main Goal

- Heading path alignment
- Eliminating offset to path

The Reference Path

- Track
  - Straight line segments
  - Waypoints
  - Parameterized curves

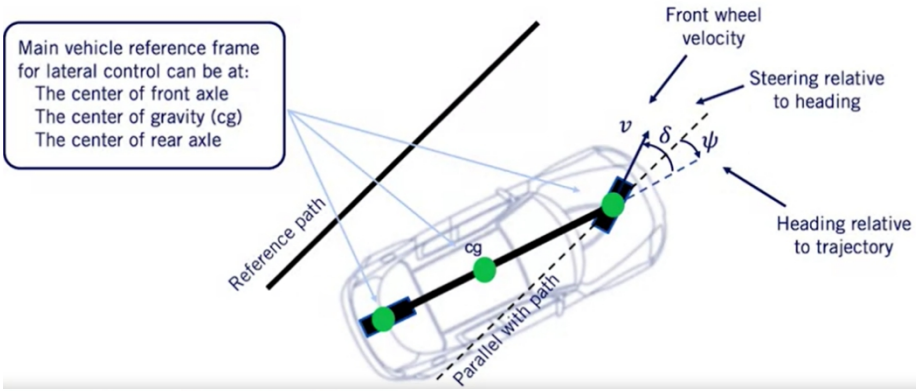


There are two types of control design:

- Geometric control (relies on geometry coordinates and Kinematics of AV)
  - Pure Pursuit controller
  - Stanley controller
- Dynamic Controller
  - Model predictive controller (MPC)
  - Feedback – Feedforward

Plant Model

- Vehicle (bicycle) model & parameters
  - All states variables and inputs defined relative to the centre of front axle

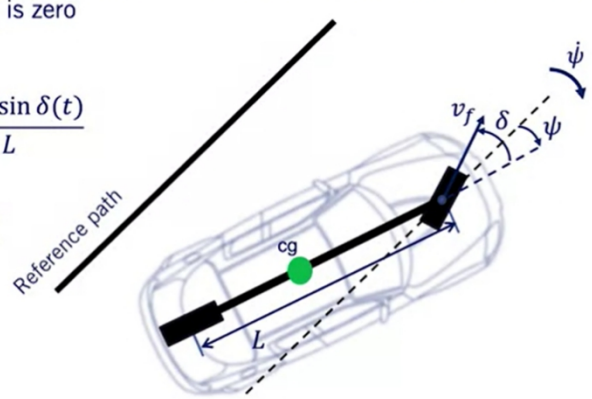


Driving Controller

- Controller error terms
  - Heading error
    - Component of velocity perpendicular to trajectory divided by ICR radius
    - Desired heading is zero

$$\dot{\psi}_{des}(t) - \dot{\psi}(t) = \frac{v_f(t)\sin\delta(t)}{L}$$

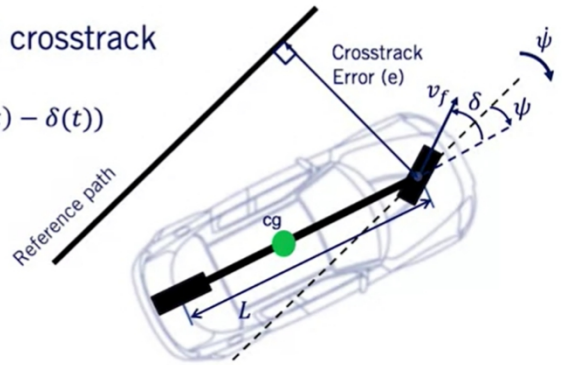
$$\dot{\psi}(t) = \frac{-v_f(t)\sin\delta(t)}{L}$$



Driving Controller

- Crosstrack error (e) :
  - Distance from center of front axle to the closest point on path
- Rate of change of crosstrack error ($\dot{e}$)

$$\dot{e}(t) = v_f(t)\sin(\psi(t) - \delta(t))$$



Geometric controller

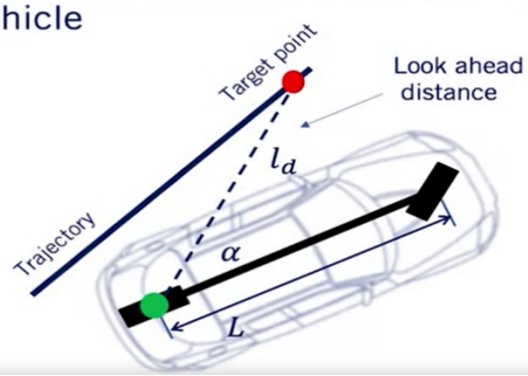
Path Tracking

- One of the most popular classes of path tracking in robotics and AV
- Tracks between Kinetic and reference geometry
- Constructs steering command
- Struggles with no slip condition
- Tire data required

Pure pursuit Controller

Pure pursuit

- Pure pursuit method consists of geometrically calculating the trajectory curvature
- Connect the centre of rear axle location to a target point on the path ahead of the vehicle



Pure pursuit - formulation

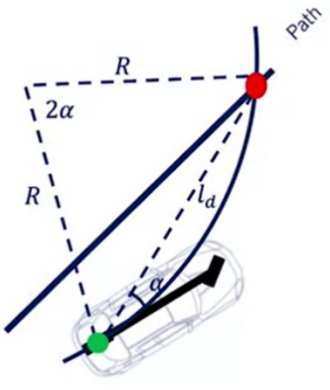
- Steering angle determined by target point location and angle between the vehicle's heading direction and lookahead direction.
- From the *law of sines*:

$$\frac{l_d}{\sin 2\alpha} = \frac{R}{\sin\left(\frac{\pi}{2} - \alpha\right)}$$

$$\frac{l_d}{2\sin\alpha\cos\alpha} = \frac{R}{\cos(\alpha)}$$

$$\frac{l_d}{\sin\alpha} = 2R$$

$$\kappa = \frac{1}{R} = \frac{2\sin\alpha}{l_d}$$ Path curvature

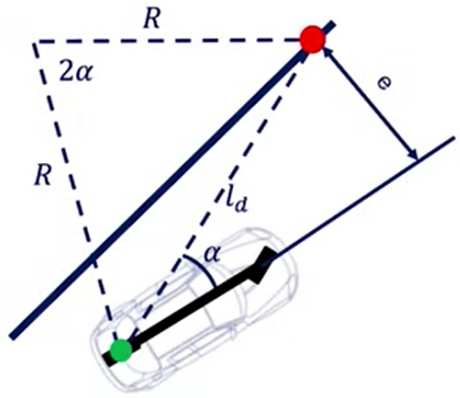


- Crosstrack error (e) is defined here as the lateral distance between the heading vector and the target point so:

$$\sin\alpha = \frac{e}{l_d}$$

$$\kappa = \frac{2\sin\alpha}{l_d}$$

$$\kappa = \frac{2}{l_d^2}e$$



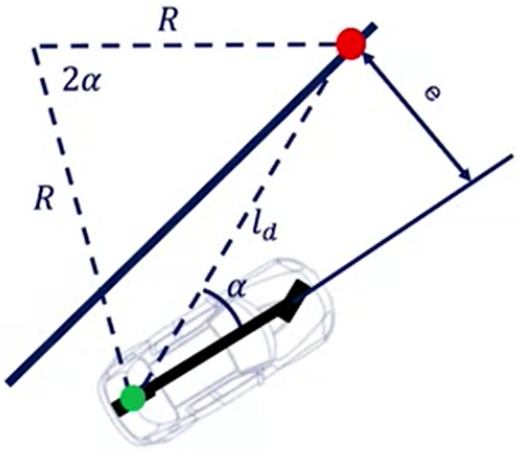
- Pure Pursuit is a proportional controller of the steering angle operating on a cross track error some look ahead distance in front of the vehicle.
- The proportional gain (2/(Ld^2)) can be tunes at different speeds (The Ld. being assigned as a function of vehicle speed).
- Lookahead (Ld.) is assigned as a linear function of vehicle:

$$l_d = Kv_f$$

$$\delta = \tan^{-1}\left(\frac{2L\sin\alpha}{l_d}\right) \qquad \kappa = \frac{2}{l_d^2}e$$

$$\delta = \tan^{-1}\left(\frac{2L\sin\alpha}{K_{dd}v_f}\right)$$

Forward velocity



Stanley Controller

- Stanley Controller uses front axle as a reference point.
- Look at both error in heading and the error in position relative to the closest point on the path.
- Define intuitive steering law to:
  - Correct heading error.
  - Correct position error.
  - Obey max steering angle.

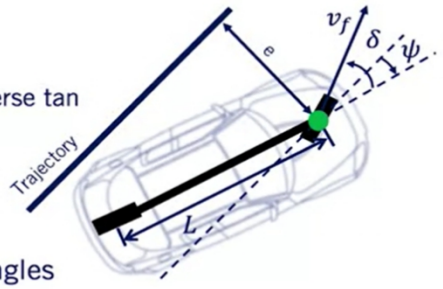
Heading control law

- Combine three requirements:
  - Steer to align heading with desired heading (proportional to heading error)

$$\delta(t) = \psi(t)$$

  - Steer to eliminate crosstrack error
    - Essentially proportional to error
    - Inversely proportional to speed
    - Limit effect for large errors with inverse tan
    - Gain k determined experimentally

$$\delta(t) = \tan^{-1}\left(\frac{ke(t)}{v_f(t)}\right)$$



  - Maximum and minimum steering angles

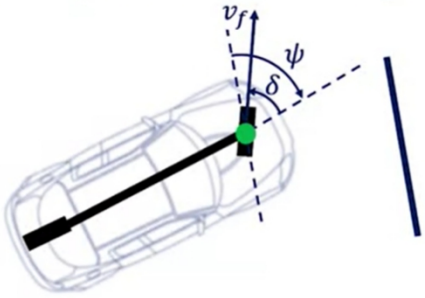
$$\delta(t) \in [\delta_{min}, \delta_{max}]$$

Combined steering law

- Stanley Control Law

$$\delta(t) = \psi(t) + \tan^{-1}\left(\frac{ke(t)}{v_f(t)}\right), \quad \delta(t) \in [\delta_{min}, \delta_{max}]$$

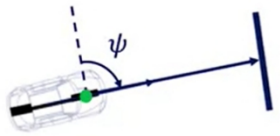
- For large heading error, steer in opposite direction
  - The larger the heading error, the larger the steering correction
  - Fixed at limit beyond maximum steering angle, assuming no crosstrack error



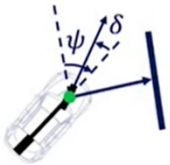
- For larger positive crosstrack error

$$\tan^{-1}\left(\frac{ke(t)}{v_f(t)}\right) \approx \frac{\pi}{2} \rightarrow \delta(t) \approx \psi(t) + \frac{\pi}{2}$$

  - As heading changes due to steering angle, the heading correction counteracts the crosstrack correction, and drives the steering angle back to zero



  - The vehicle approaches the path, crosstrack error drops, and steering command starts to correct heading alignment.



Error Dynamics

- The error dynamics when not at maximum steering angle are:

$$\dot{e}(t) = -v_f(t)\sin(\psi(t) - \delta(t)) = -v_f(t)\sin\left(\tan^{-1}\left(\frac{ke(t)}{v_f(t)}\right)\right)$$

$$= \frac{-ke(t)}{\sqrt{1 + \left(\frac{ke(t)}{v_f}\right)^2}}$$

- For small crosstrack errors, leads to exponential decay characteristics

$$\dot{e}(t) \approx -ke(t)$$

- Stanley Controller works with two constrains:
  - Large initial cross-track error
  - Large initial heading error
- Control initiation works in the following sequence:
  - Correct heading error > Correct cross-track error > Correct path

Limitations

- Difficult in low-speed operations
  - As velocity is inverse function, it gets swingy at low velocity due to noisy velocity data
- To rectify this issue, we can make it a PD controller to reduce noise and improve output
- We can also inset a feedforward loop into the mix.

Dynamic modelling

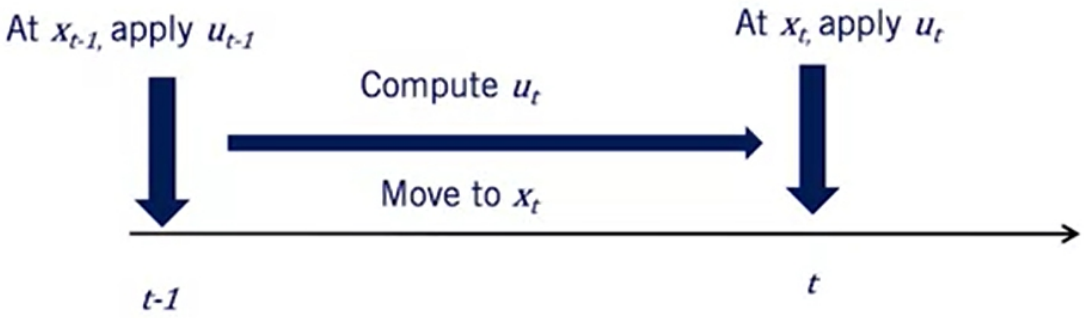
Model Predictive Control (MPC)

- Numerically solving an optimization at each time step
- Receding horizon approach

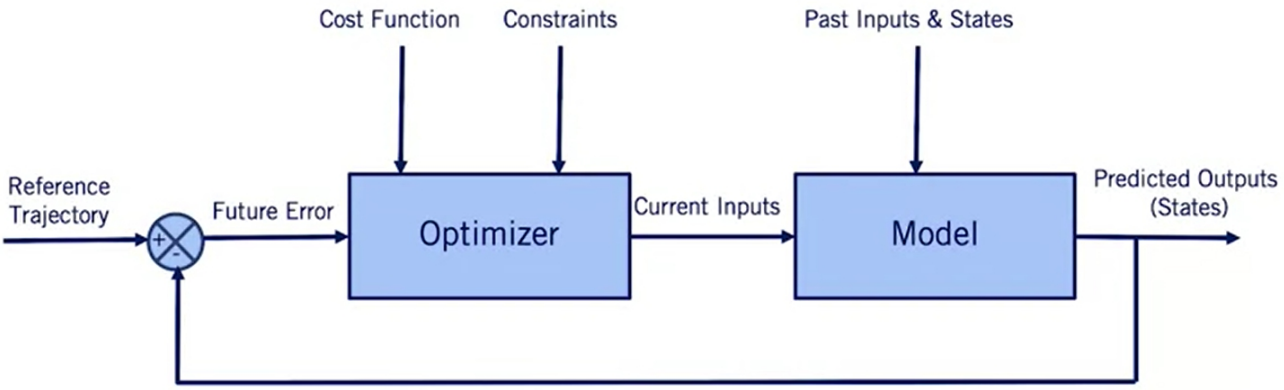
- Advantages
  - Straight forward formulation
  - Explicitly Handles approach
  - Applicable to linear or non-linear model
- Disadvantages
  - Computationally expensive
- Concept / Algorithm
  - Concepts optimization on each time step use predictive optimization to save time and computation.

Receding horizon Control

- Receding Horizon Control Algorithm
  - Pick receding horizon length (T)
  - For each time step, t
  - Set initial state to predicted state, x_t
    - Perform optimization over finite horizon t to T while traveling from x_{t-1} to x_t
    - Apply first control command, u_t, from optimization at time t



MPC structure



Linear MPC formulation

- Linear time-invariant discrete time model:

$$x_{t+1} = Ax_t + Bu_t$$

- MPC seeks to find control policy U

$$U = \{u_{t|t}, u_{t+1|t}, u_{t+2|t}, \dots\}$$

- Objective function - regulation:

$$J(x(t), U) = \sum_{j=t}^{t+T-1} x_{j|t}^T Q x_{j|t} + u_{j|t}^T R u_{j|t}$$

- Objective function - tracking:

$$\delta x_{j|t} = x_{j|t,des} - x_{j|t} \qquad J(x(t), U) = \sum_{j=t}^{t+T-1} \delta x_{j|t}^T Q \delta x_{j|t} + u_{j|t}^T R u_{j|t}$$

Linear MPC SOLUTION

- Unconstrained, finite horizon, discrete time problem formulation:

$$\min_{U \triangleq \{u_{t|t}, u_{t+1|t}, \dots\}} J(x(t), U) = x_{t+T|t}^{\mathrm{T}} \, Q_f \, x_{t+T|t} + \sum_{j=t}^{t+T-1} x_{j|t}^{\mathrm{T}} Q x_{j|t} + u_{j|t}^{\mathrm{T}} R u_{j|t}$$

$$s.t. \qquad x_{j+1|t} = A x_{t|t} + B u_{t|t}\,, \qquad t \le j \le t+T-1$$

- Linear quadratic regulator, provides a closed form solution
  - Full state feedback: $u_t = -Kx_t$
  - Control gain K is a matrix
  - Refer to supplemental materials

(Non)Linear MPC formulation

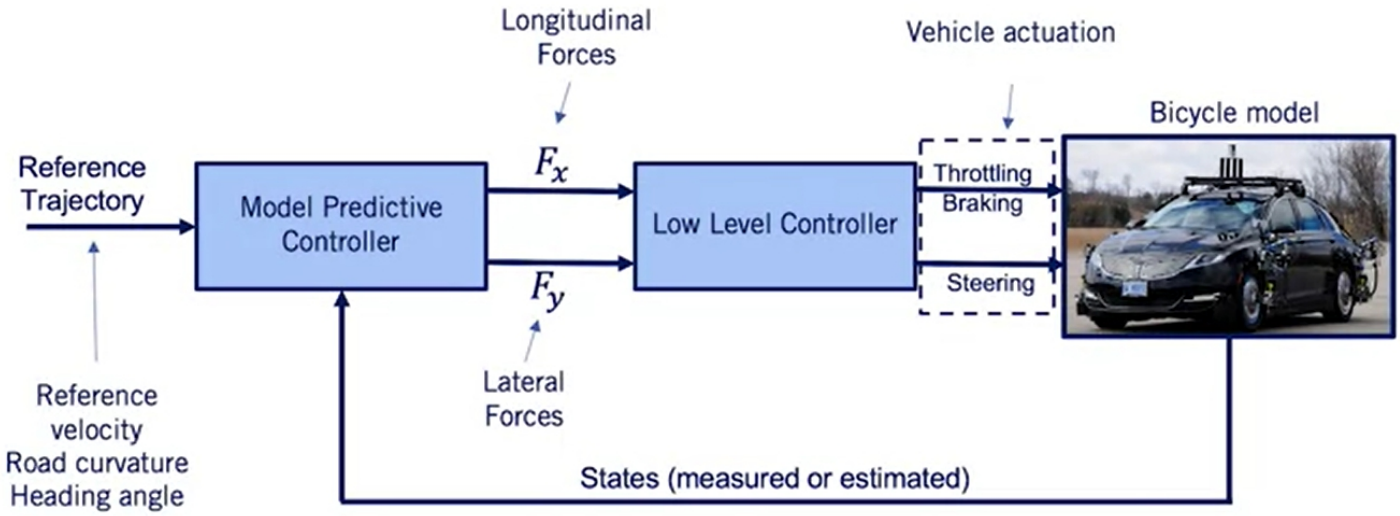
- Constrained (non)linear finite horizon discrete time case

$$\min_{U \triangleq \{u_{t|t}, u_{t+1|t}, \dots\}} J(x(t), U) = \sum_{j=t}^{t+T} C(x_{j|t}, u_{j|t})$$

$$\begin{aligned}
s.t. \quad & x_{j+1|t} = f\left(x_{j|t}, u_{j|t}\right), && t \le j \le t+T-1 \\
& x_{min} \le x_{j+1|t} \le x_{max}\,, && t \le j \le t+T-1 \\
& u_{min} \le u_{j|t} \le u_{max}\,, && t \le j \le t+T-1 \\
& g\left(x_{j|t}, u_{j|t}\right) \le 0, && t \le j \le t+T-1 \\
& h\left(x_{j|t}, u_{j|t}\right) = 0, && t \le j \le t+T-1
\end{aligned}$$

- No closed form solution, must be solved numerically

Vehicle Lateral Control



- With model predictive control, new state is achieved with every at each time step.

Cost Function minimization

- Deviation from desired trajectory
- Minimization of control command magnitude

Constraints – Subject to

- Longitudinal and lateral dynamic models
- Tire force limits

Can incorporate low level controller, adding constraints for

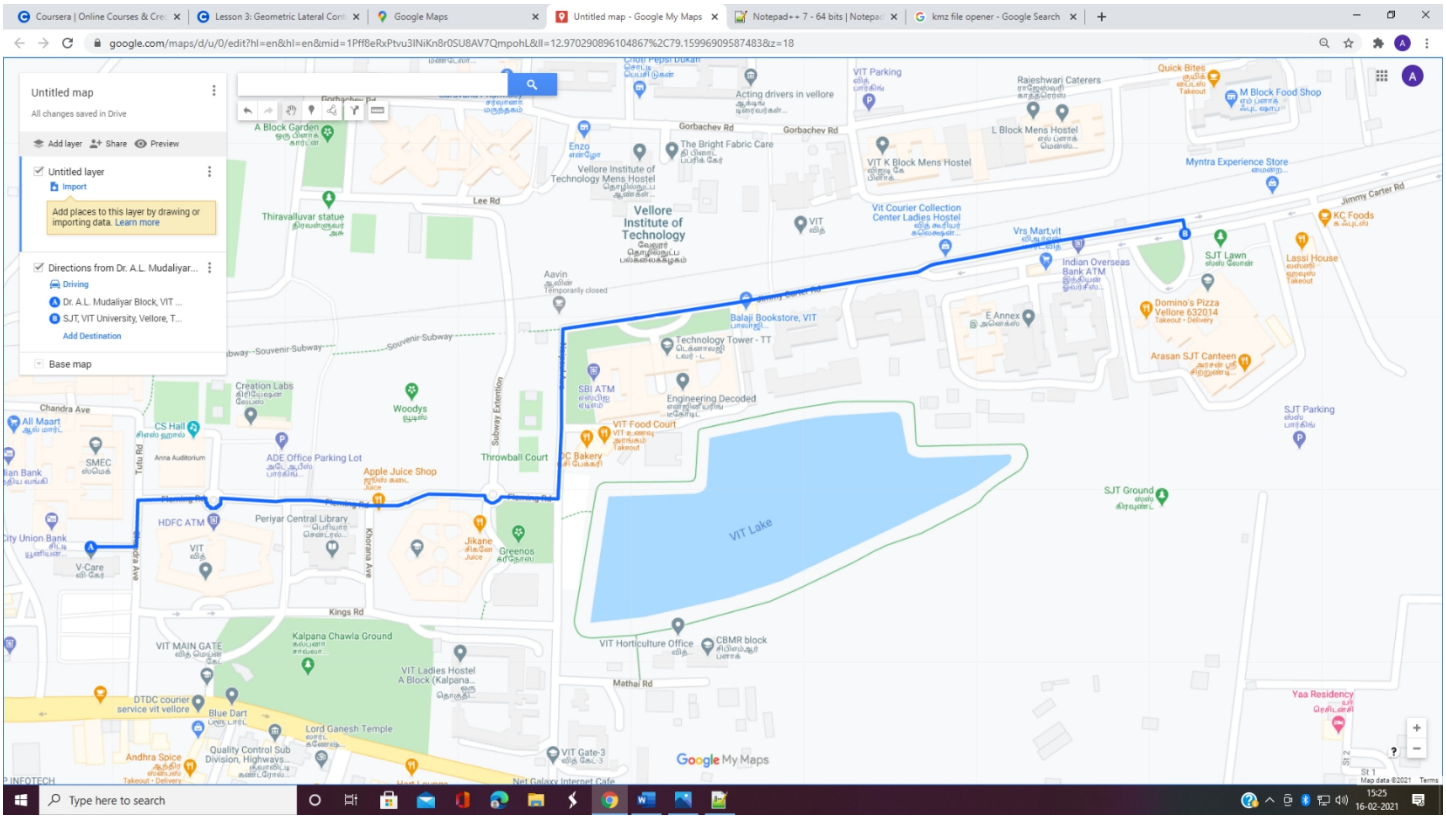
- Full dynamic vehicle model
- Actuator model
- Tire force model

This flexibility and convenience come at the cost of increased computational requirements and relies on the availability of robust optimization solvers to always return feasible solutions in the available time window.

5. Reference Path

After conferring with our Project Guide Professor C Ramesh Kumar, we have selected a reference path within the VIT campus to test Vehicle perception, State Estimation, Motion planning and localization.

The path chosen is from VIT Autonomous Vehicle Research lab (V-Care) to SJT building.



For This path we were able to get the exact coordinates in latitudes and longitudes from point A to B (A – V-care, B - SJT building)

Coordinates are listed below for the path shown above.

79.15495,12.96932,0
79.15531,12.96932,0
79.15532,12.96968,0
79.15586,12.9697,0
79.15586,12.96969,0
79.15586,12.96968,0
79.15586,12.96967,0
79.15586,12.96966,0
79.15587,12.96966,0
79.15587,12.96965,0
79.15587,12.96964,0
79.15588,12.96964,0
79.15589,12.96963,0
79.1559,12.96963,0
79.1559,12.96962,0
79.15591,12.96962,0
79.15592,12.96962,0
79.15593,12.96962,0
79.15594,12.96962,0
79.15594,12.96963,0
79.15595,12.96963,0
79.15596,12.96963,0
79.15596,12.96964,0
79.15597,12.96964,0
79.15597,12.96965,0
79.15598,12.96965,0
79.15598,12.96966,0
79.15598,12.96967,0
79.15598,12.96968,0
79.15598,12.96969,0
79.15598,12.9697,0
79.15642,12.96967,0
79.1565,12.96968,0
79.15669,12.96968,0
79.15716,12.96964,0
79.15736,12.96964,0
79.15739,12.96964,0
79.15743,12.96965,0
79.15745,12.96966,0
79.15747,12.96966,0
79.1575,12.96968,0
79.15757,12.96971,0
79.1576,12.96972,0
79.15763,12.96973,0
79.15765,12.96973,0
79.15769,12.96973,0
79.15801,12.96972,0
79.15808,12.96972,0
79.15808,12.96971,0
79.15808,12.9697,0
79.15809,12.9697,0
79.15809,12.96969,0
79.15809,12.96968,0

Controller code

```
import cutils
import numpy as np
# from sympy import integrate
import math
Kp = 2.0        # speed proportional gain
Ki = 0.05
Kd = 0.01
k = 0.1         # look forward gain
Lfc = 1.0       # look-ahead distance
L = 2.9
class Controller2D(object):
    def __init__(self, waypoints):
        self.vars = cutils.CUtils()
        self._current_x = 0
        self._current_y = 0
        self._current_yaw = 0
        self._current_speed = 0
        self._desired_speed = 0
        self._current_frame = 0
        self._current_timestamp = 0
        self._start_control_loop = False
        self._set_throttle = 0
        self._set_brake = 0
        self._set_steer = 0
        self._waypoints = waypoints
        self._conv_rad_to_steer = 180.0 / 70.0 / np.pi
        self._pi = np.pi
        self._2pi = 2.0 * np.pi
    def update_values(self, x, y, yaw, speed, timestamp, frame):
        self._current_x = x
        self._current_y = y
        self._current_yaw = yaw
        self._current_speed = speed
```

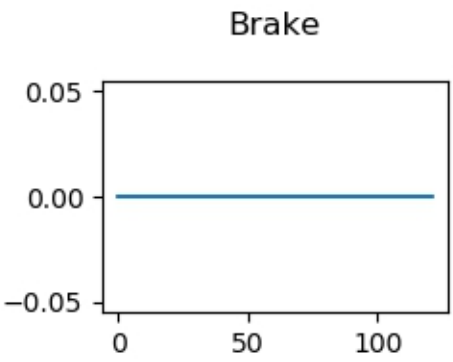
Deployment code

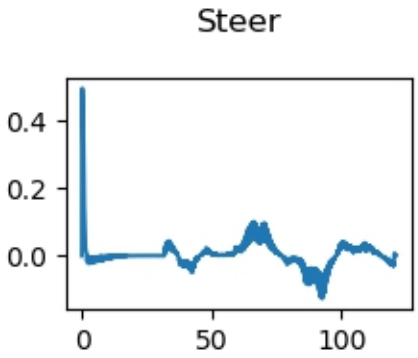
```
"""
CARLA waypoint follower assessment client script.
A controller assessment to follow a given trajectory, where the trajectory
can be defined using way-points.
"""
from __future__ import print_function
from __future__ import division
# System level imports
import sys
import os
import argparse
import logging
import time
import math
import numpy as np
import csv
import matplotlib.pyplot as plt
import controller2d
import configparser
# Script level imports
sys.path.append(os.path.abspath(sys.path[0] + '/..'))
import live_plotter as lv  # Custom live plotting library
from carla import sensor
from carla.client import make_carla_client, VehicleControl
from carla.settings import CarlaSettings
from carla.tcp import TCPConnectionError
from carla.controller import utils
"""
Configurable params
"""
```

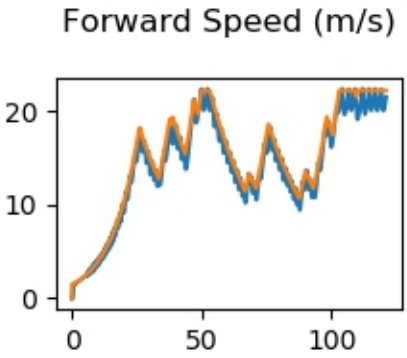
State Estimation code

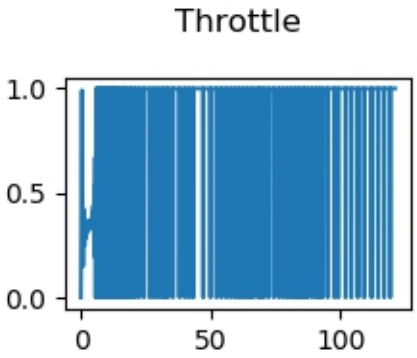
```
import pickle

import numpy as np

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

from rotations import angle_normalize,
rpy_jacobian_axis_angle, skew_symmetric,
Quaternion

from numpy.linalg import inv

import rotations

with open('data/pt1_data.pkl', 'rb') as file:

    data = pickle.load(file)

# with open('data/pt3_data.pkl', 'rb') as file:

#     data = pickle.load(file)

# Each element of the data dictionary is stored as
an item from the data dictionary, which we

# will store in local variables, described by the
following:

#   gt: Data object containing ground truth. with
the following fields:

#     a: Acceleration of the vehicle, in the inertial
frame

#     v: Velocity of the vehicle, in the inertial frame

#     p: Position of the vehicle, in the inertial frame

#     alpha: Rotational acceleration of the vehicle,
in the inertial frame

#     w: Rotational velocity of the vehicle, in the
inertial frame

#     r: Rotational position of the vehicle, in Euler
(XYZ) angles in the inertial frame

#     _t: Timestamp in ms.

#   imu_f: StampedData object with the imu
specific force data (given in vehicle frame).

#     data: The actual data

#     t: Timestamps in ms.

#   imu_w: StampedData object with the imu
rotational velocity (given in the vehicle frame).

#     data: The actual data

#     t: Timestamps in ms.

#   gnss: StampedData object with the GNSS
```

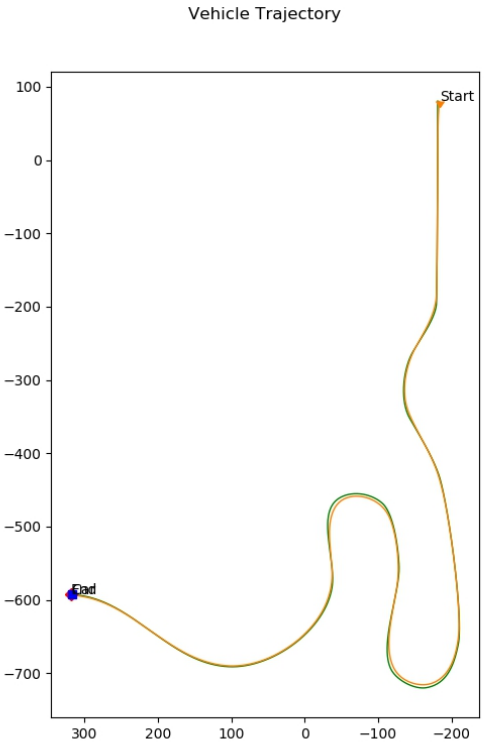
Analysis


Brake
0.05
0.00
−0.05
0
50
100

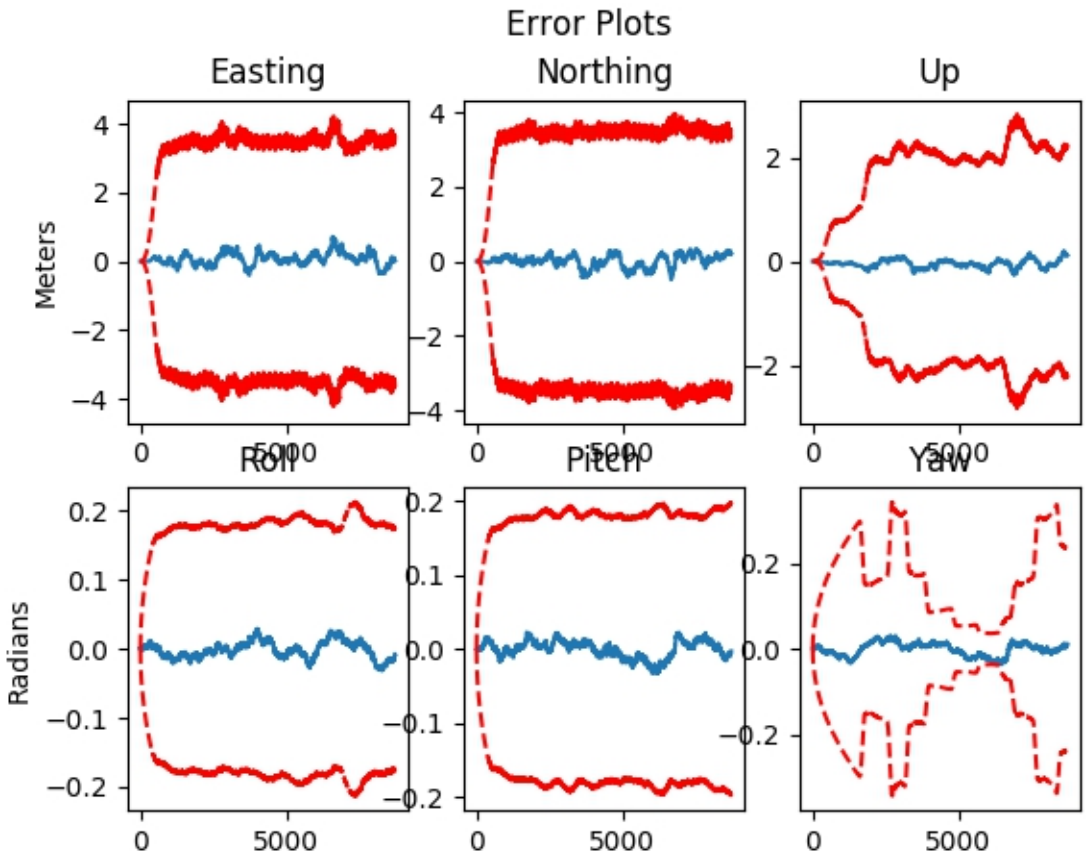

Steer
0.4
0.2
0.0
0
50
100

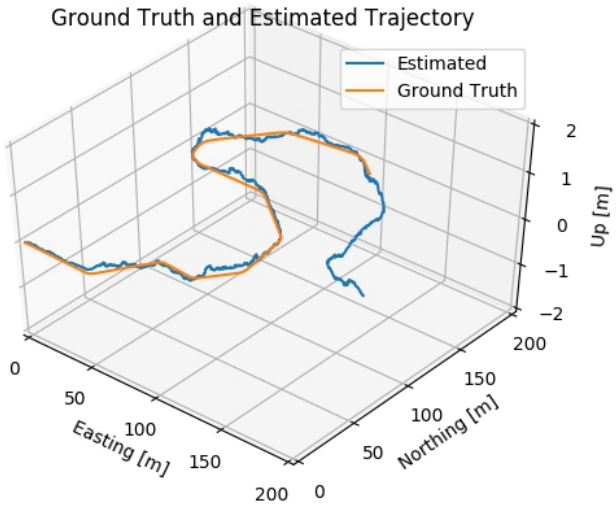

Forward Speed (m/s)
20
10
0
0
50
100


Throttle
1.0
0.5
0.0
0
50
100

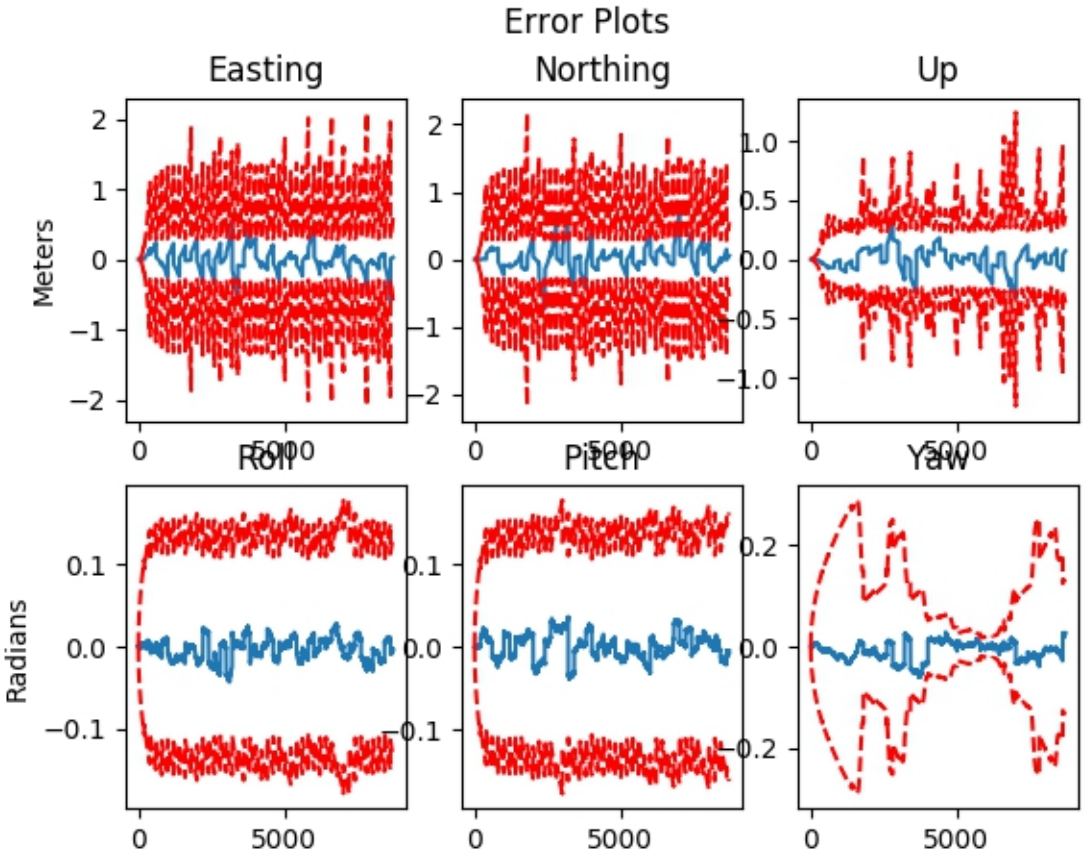

Vehicle Trajectory
Start
End
100
0
−100
−200
−300
−400
−500
−600
−700
300
200
100
0
−100
−200

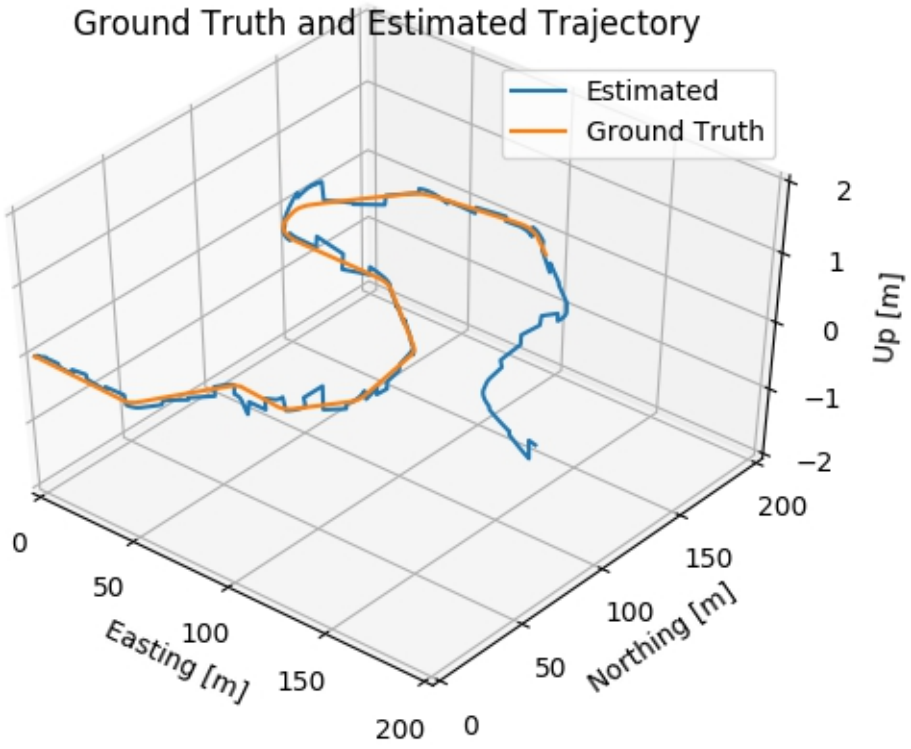
Implementation of Extended Kalman Filter



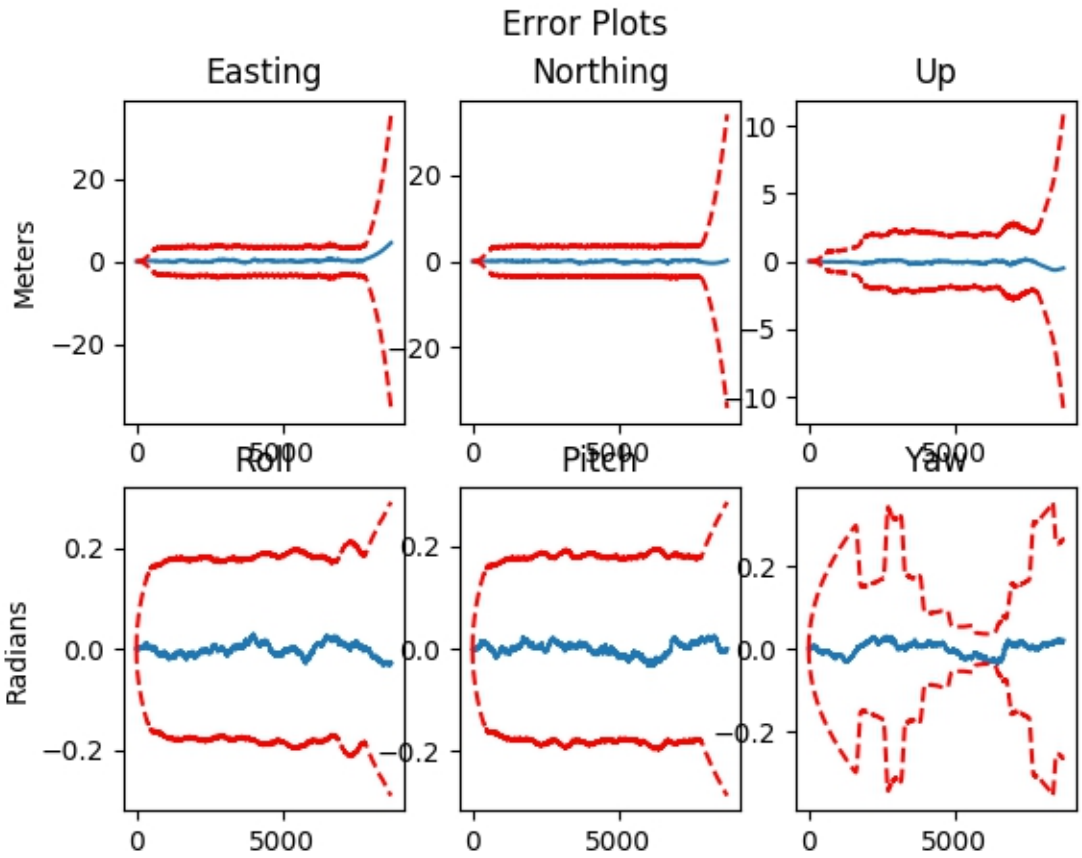


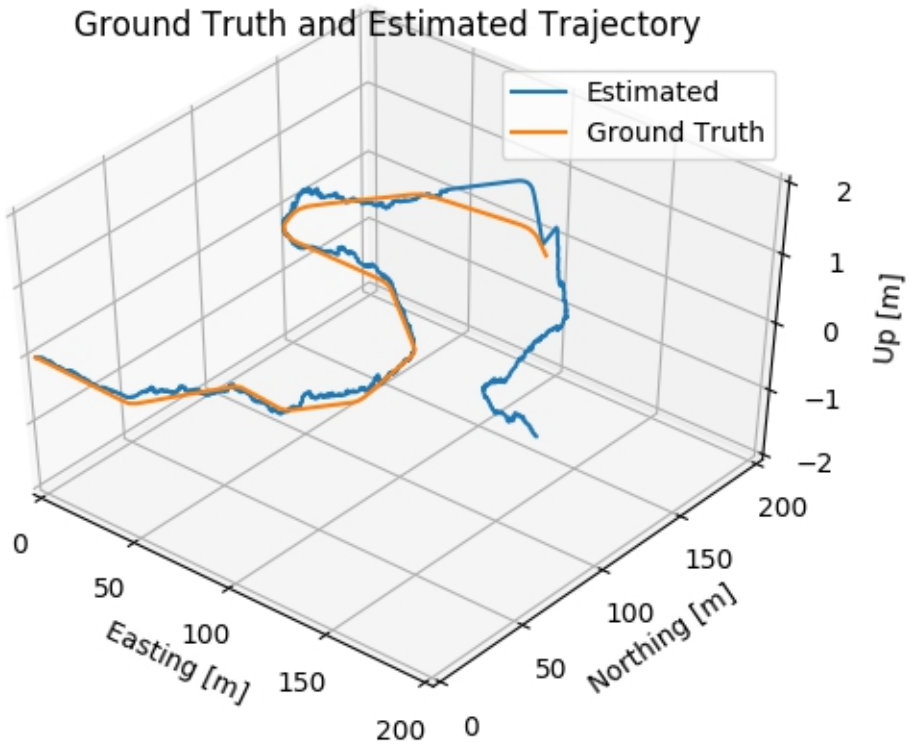
Effects of sensor miscalibration





Effects of sensor dropout





6. Ideation

The Idea is to implement an Integrated control system within the vehicle that can complete this task Autonomously and gather crucial data that will directly contribute to further research in a generalized autonomous vehicle technology.

 # All THE WORK WILL BE DONE IN MATLAB AUTOMATED DRIVING TOOLBOX AND SIMULINK.

The project can be approached in two different ways:

PID + Feedforward + Stanley controller

Stanley controller for longitudinal control with Proportion and integral control

Input:

Reference Velocity - Reference velocity, in meters per second, specified as a real scalar.

Current Velocity - Current velocity of the vehicle, in meters per second, specified as a real scalar.

Driving Direction - Driving direction of vehicle, specified as 1 for forward motion and -1 for reverse motion.

Reset — Trigger to reset integral of velocity error - Trigger to reset the integral of velocity error, $e(k)$, to zero. A value of 0 holds $e(k)$ steady. A nonzero value resets $e(k)$.

Output:

Acceleration command: Acceleration command, returned as a real scalar in the range [0, M_A], where M_A is the value of the **Maximum longitudinal acceleration (m/s^2)** parameter.

Deceleration command: Deceleration command, returned as a real scalar in the range [0, M_D], where M_D is the value of the **Maximum longitudinal deceleration (m/s^2)** parameter.

Parameters:

Proportional Gain, Kp: Proportional gain of controller, K_p, specified as a positive real scalar.

Integral Gain Ki: Integral gain of controller, K_i, specified as a positive real scalar.

Sample time: Sample time of controller, in seconds, specified as a positive real scalar.

Maximum longitudinal acceleration: Maximum longitudinal acceleration, in meters per second squared, specified as a positive real scalar. The block saturates the output from the **AccelCmd** to the range [0, M_A], where M_A is the value of this parameter. Values above M_A are set to M_A.

Maximum Longitudinal Deceleration: Maximum longitudinal deceleration, in meters per second squared, specified as a positive real scalar. The block saturates the output from the **DecelCmd** port to the range [0, M_D], where M_D is the value of this parameter. Values above M_D are set to M_D.

The above-mentioned model will account for the state estimation of the vehicle and calculated full longitudinal vehicle control in terms of throttle position

The Mathematics behind the concept

The Longitudinal Controller Stanley block implements a discrete proportional-integral (PI) controller with integral anti-windup, as described by the Anti-windup method (Simulink) parameter of the PID Controller block. The block uses this equation:

$$u(k) = \left(K_p + K_i \frac{T_s}{z - 1}\right) e(k)$$

- $u(k)$ is the control signal at the kth time step.
- K_p is the proportional gain, as set by the **Proportional gain, Kp** parameter.
- K_i is the integral gain, as set by the **Integral gain, Ki** parameter.
- T_s is the sample time of the block in seconds, as set by the **Sample time (s)** parameter.
- $e(k)$ is the velocity error (**CurrVelocity** – **RefVelocity**) at the kth time step. For each k, this error is equal to the difference between the current velocity and reference velocity inputs (**CurrVelocity** – **RefVelocity**).

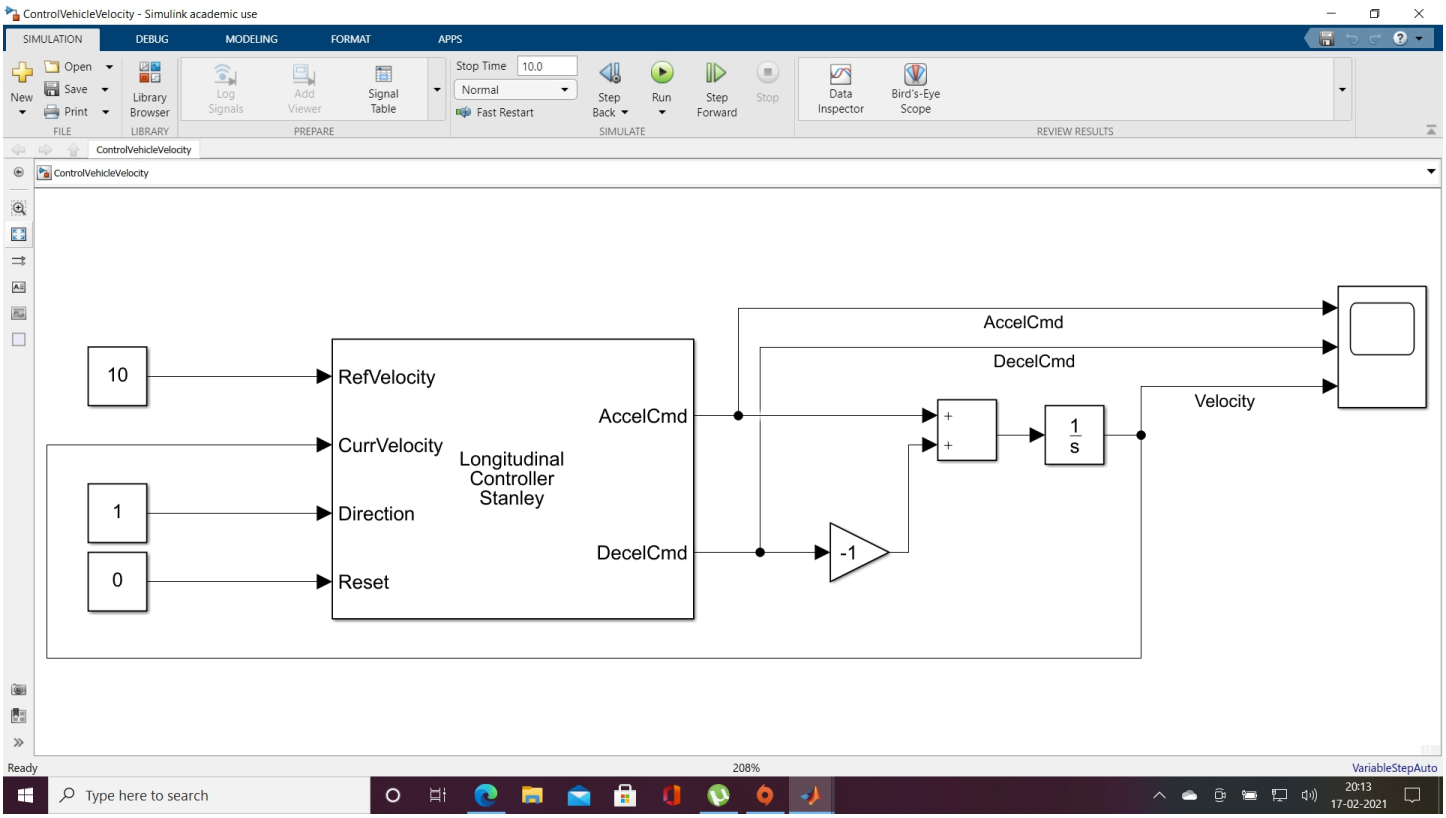
The control signal, u, determines the value of acceleration command **AccelCmd** and deceleration command **DecelCmd**. The block saturates the acceleration and deceleration commands to respective ranges of [0, M_A] and [0, M_D], where:

- M_A is value of the **Maximum longitudinal acceleration (m/s^2)** parameter.
- M_D is the value of the **Maximum longitudinal deceleration (m/s^2)** parameter.

At each time step, only one of the **AccelCmd** and **DecelCmd** port values is positive, and the other port value is 0. In other words, the vehicle can either accelerate or decelerate in one time step, but it cannot do both at one time.

The direction of motion, as specified in the **Direction** input port, determines which command is positive at the given time step.

Direction Port Value	Control Signal Value $u(k)$	AccelCmd Port Value	DecelCmd Port Value	Description
1 (forward motion)	$u(k) > 0$	positive real scalar	0	Vehicle speeds up as it travels forward
	$u(k) < 0$	0	positive real scalar	Vehicle slows down as it travels forward
-1 (reverse motion)	$u(k) > 0$	0	positive real scalar	Vehicle slows down as it travels in reverse
	$u(k) < 0$	positive real scalar	0	Vehicle speeds up as it travels in reverse



This model was created on for this specific task, and will be further connected to the Curtis controller available in the lab. This geometric path tracking and real-time Proportion Integral formulation can give a vey accurate data to work with and reduce sensor signal noise significantly.

As we can see the above Simulink model mimics the previously explained longitudinal vehicle control with PID and Feedforward loop, and bares with it all its advantages and limitations.

With the above-mentioned coordinates can be integrated in the above system where we can calculate the reference velocity by change in each coordinate with respect to time, the current velocity keeps changing as the surroundings change and accelerate and decelerate with the autonomous system requirements.

Stanley Controller for lateral control with kinematic bicycle model and dynamic bicycle model

Input:

Reference position: Reference pose, specified as an [*x*, *y*, *Θ*] vector. *x* and *y* are in meters, and *Θ* is in degrees. *x* and *y* specify the reference point to steer the vehicle toward. *Θ* specifies the orientation angle of the path at this reference point and is positive in the counterclockwise direction.

Current Position: Current pose of the vehicle, specified as an [*x*, *y*, *Θ*] vector. *x* and *y* are in meters, and *Θ* is in degrees.*x* and *y* specify the location of the vehicle, which is defined as the centre of the vehicle's rear axle. *Θ* specifies the orientation angle of the vehicle at location (*x*, *y*) and is positive in the counter clockwise direction.

Current longitudinal velocity: Current longitudinal velocity of the vehicle, specified as a real scalar. Units are in meters per second.

- If the vehicle is in forward motion, then this value must be greater than 0.
- If the vehicle is in reverse motion, then this value must be less than 0.
- A value of 0 represents a vehicle that is not in motion.

Driving direction of vehicle: Driving direction of the vehicle, specified as 1 for forward motion or -1 for reverse motion. The driving direction determines the position error and angle error used to compute the steering angle command.

Curvature of path: Curvature of the path at the reference point, in radians per meter, specified as a real scalar.

Current Yaw rate: Current yaw rate of the vehicle, in degrees per second, specified as a real scalar. The current yaw rate is the rate of change in the angular velocity of the vehicle.

Current Steering angle: Current steering angle of the vehicle, in degrees, specified as a real scalar. This value is positive in the counterclockwise direction.

Output:

Steering Angle command: Steering angle command, in degrees, returned as a real scalar. This value is positive in the counterclockwise direction.

Parameters:

Vehicle model: Select the type of vehicle model to set the Stanley method control law used by the block.

- Kinematic bicycle model — Kinematic bicycle model for path following in low-speed environments such as parking lots, where inertial effects are minimal
- Dynamic bicycle model — Dynamic bicycle model for path following in high-speed environments such as highways, where inertial effects are more pronounced

Position Gain Of vehicle in forward motion: Position gain of the vehicle when it is in forward motion, specified as a positive scalar. This value determines how much the position error affects the steering angle. Typical values are in the range [1, 5]. Increase this value to increase the magnitude of the steering angle.

Position Gain Of vehicle in forward motion: Position gain of the vehicle when it is in reverse motion, specified as a positive scalar. This value determines how much the position error affects the steering angle. Typical values are in the range [1, 5]. Increase this value to increase the magnitude of the steering angle.

Yaw rate feedback gain: Yaw rate feedback gain, specified as a nonnegative real scalar. This value determines how much weight is given to the current yaw rate of the vehicle when the block computes the steering angle command.

Steering Angle feedback gain: Steering angle feedback gain, specified as a nonnegative real scalar. This value determines how much the difference between the current steering angle command, **SteerCmd**, and the current steering angle, **CurrSteer**, affects the next steering angle command.

Wheelbase of vehicle: Distance between the front and rear axle of the vehicle, in meters, specified as a real scalar.

Vehicle Mass: Vehicle mass, in kilograms, specified as a positive real scalar.

Distance from centre of mass to front axle: Longitudinal distance from the vehicle's center of mass to its front wheel axle, in meters, specified as a positive real scalar.

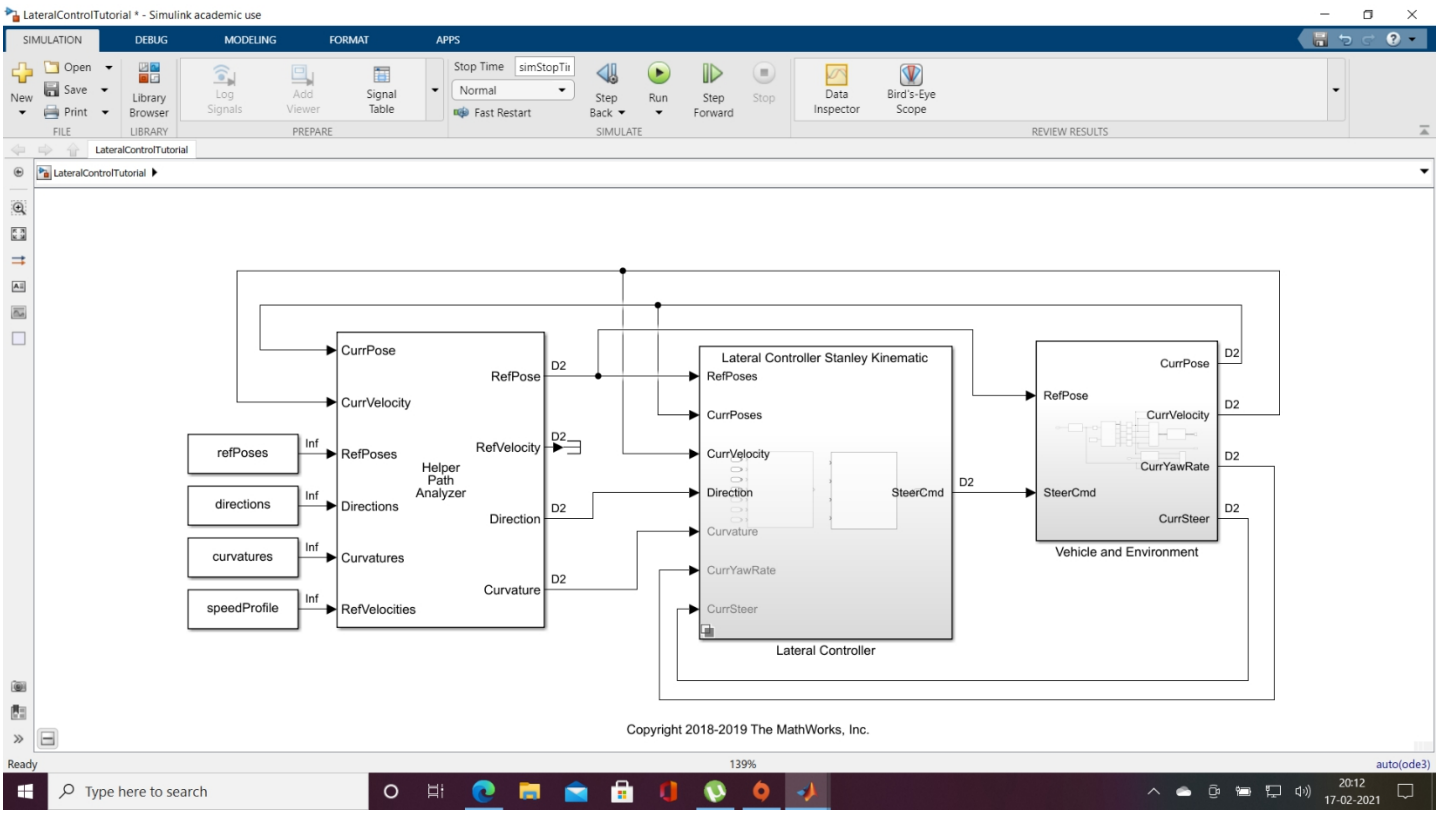
Distance from centre of mass to rear axle: Longitudinal distance from the vehicle's center of mass to its rear wheel axle, in meters, specified as a positive real scalar.

Cornering Stiffness of front tires: Cornering stiffness of front tires, in Newtons per radian, specified as a positive real scalar.

Maximum allowed steering angle: Maximum allowed steering angle of the vehicle, in degrees, specified as a real scalar in the range (0, 180).

The output from the **SteerCmd** port is saturated to the range [–*M*, *M*], where *M* is the value of the **Maximum steering angle (deg)** parameter.

- Values below –*M* are set to –*M*.
- Values above *M* are set to *M*.



As visible from the above control diagram, with all the inputs and the parameters we can optimally calculate a steering angle vie a geometric lateral controller.

We use the same principle used in the longitudinal vehicle controller, we will integrate the above-mentioned coordinates in the lateral control system where it constantly calculates a reference position, calculate and reduce heading, position and cross track error to current position this calculating a very sound lateral control system which can be tested on the AV being built in the V-CARE facility.

Now, Both the Longitudinal and lateral vehicle controllers will interact with each other constantly as the both need to work in sync to actually Implement an AV control system.

The major drawback with Stanley controller is it reacts differently in low and high velocity conditions and can be unpredictable and inaccurate, although absolute integral error with respect to the reference path is <0.4, but this fraction can be the difference between close calls as the machinery is responsible for life in form of passengers, this can be Ofsted by using a **Kalman Filter** to filter the noisy data and get outstanding results.

Model Predictive Control

Adaptive cruise control System

A vehicle (ego car) equipped with adaptive cruise control (ACC) has a sensor, such as radar, that measures the distance to the preceding vehicle in the same lane (lead car), D_{rel}. The sensor also

measures the relative velocity of the lead car, V_{rel}. The ACC system operates in the following two modes:

- Speed control: The ego car travels at a driver-set speed.
- Spacing control: The ego car maintains a safe distance from the lead car.

The ACC system decides which mode to use based on real-time radar measurements. For example, if the lead car is too close, the ACC system switches from speed control to spacing control. Similarly, if the lead car is further away, the ACC system switches from spacing control to speed control. In other words, the ACC system makes the ego car travel at a driver-set speed as long as it maintains a safe distance.

Input:

AV velocity setpoint: Ego vehicle velocity setpoint in m/s. When there is no lead vehicle, the controller tracks this velocity.

Safe time gap: Safe time gap in seconds between the lead vehicle and the ego vehicle. This time gap is used to calculate the minimum safe following distance constraint.

Ego Vehicle velocity: Ego vehicle velocity in m/s.

Distance between ego vehicle and lead vehicle: Distance in meters between lead vehicle and ego vehicle. To calculate this signal, subtract the ego vehicle position from the lead vehicle position.

Velocity difference between lead vehicle and ego vehicle: Velocity difference in meters per second between lead vehicle and ego vehicle. To calculate this signal, subtract the ego vehicle velocity from the lead vehicle velocity.

Minimum ego vehicle acceleration: Minimum ego vehicle longitudinal acceleration constraint in m/s^2. Use this input port when the minimum acceleration varies at run time.

Maximum ego vehicle acceleration: Maximum ego vehicle longitudinal acceleration constraint in m/s^2. Use this input port when the maximum acceleration varies at run time.

Controller optimization enable signal: Controller optimization enable signal. When this signal is:

- Nonzero, the controller performs optimization calculations and generates a **Longitudinal acceleration** control signal.
- Zero, the controller does not perform optimization calculations. In this case, the **Longitudinal acceleration** output signal remains at the value it had when the optimization was disabled. The controller continues to update its internal state estimates.

Longitudinal acceleration applied to ego vehicle: Actual longitudinal acceleration in m/s^2 applied to the ego vehicle. The controller uses this signal to estimate the ego vehicle model states. Use this input port when the control signal applied to the ego vehicle does not match the optimal control signal computed by the model predictive controller. This mismatch can occur when, for example:

- The Adaptive Cruise Control System is not the active controller. Maintaining an accurate state estimate when the controller is not active prevents bumps in the control signal when the controller becomes active.
- The acceleration actuator fails and does not provide the correct control signal to the ego vehicle.

Output:

Acceleration control signal: Acceleration control signal in m/s^2 generated by the controller.

Parameters:

Ego vehicle model: The linear model from the ego vehicle longitudinal acceleration to its longitudinal velocity, specified as an LTI model or a linear System Identification Toolbox model. The controller creates its internal predictive model by augmenting the ego vehicle dynamic model.

Initial velocity of ego vehicle: Initial velocity in m/s of the ego vehicle model, which can differ from the actual ego vehicle initial velocity.

Minimum spacing to lead vehicle: Minimum spacing in meters between the lead vehicle and the ego vehicle. This value corresponds to the target relative distance between the ego and lead vehicles when the ego vehicle velocity is zero.

Maximum Longitudinal velocity: Maximum ego vehicle longitudinal velocity in m/s.

Adaptive Cruise Controller Constraints

Minimum ego vehicle acceleration: Minimum ego vehicle longitudinal acceleration constraint in m/s^2.

If the minimum acceleration varies over time, add the **Minimum longitudinal acceleration** input port to the block by selecting **Use external source**.

Maximum ego vehicle acceleration: Maximum ego vehicle longitudinal acceleration constraint in m/s^2.

If the maximum acceleration varies over time, add the **Maximum longitudinal acceleration** input port to the block by selecting **Use external source**.

Predictive controller settings

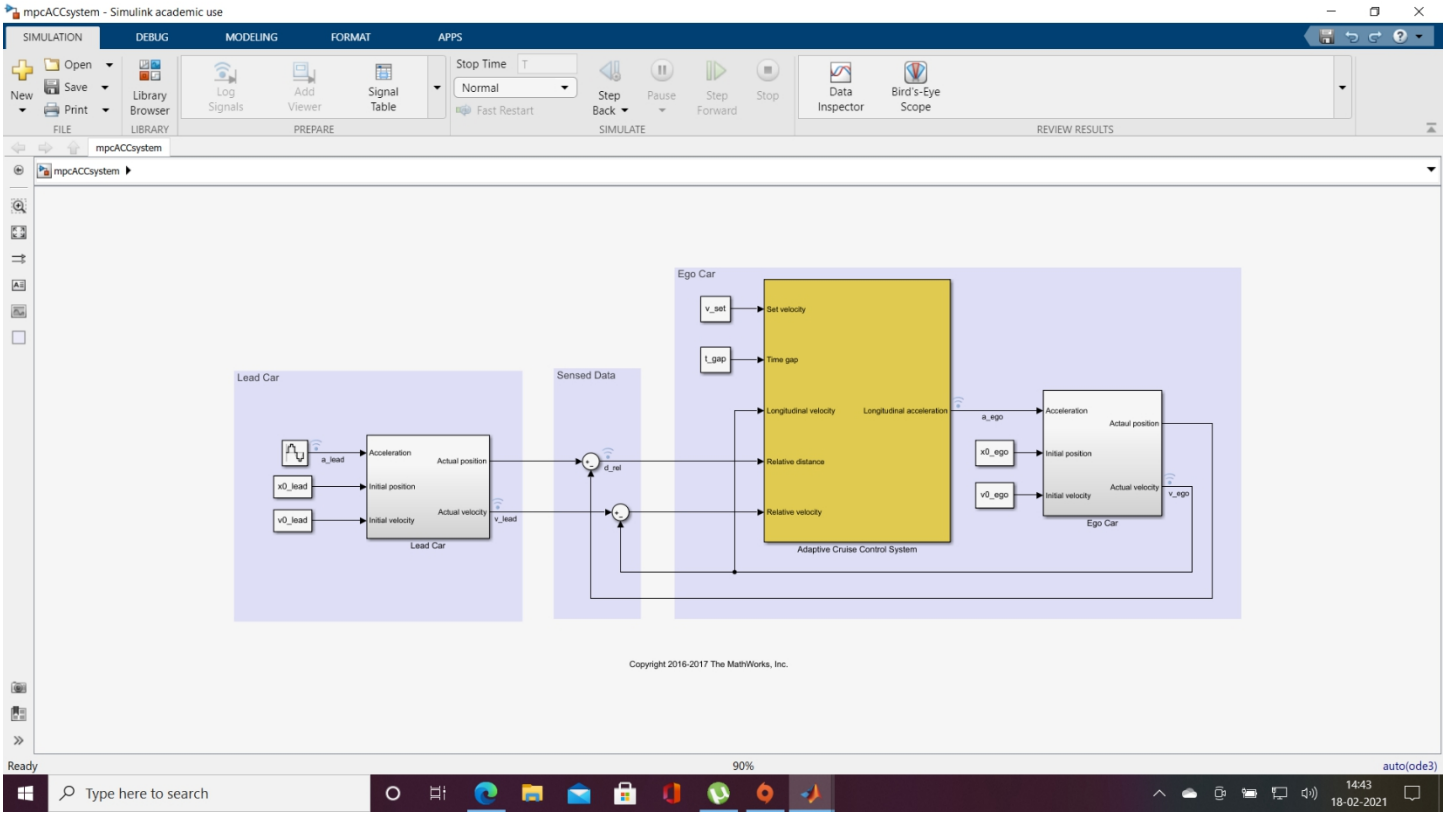
Controller sample time: Controller sample time in seconds.

Controller prediction horizon: Controller prediction horizon steps. The controller prediction time is the product of the sample time and the prediction horizon.

Closed loop controller performance: Closed-loop controller performance. The default parameter value provides a balanced controller design. Specifying a:

- Smaller value produces a more robust controller with smoother control actions.
- Larger value produces a more aggressive controller with a faster response time.

When you modify this parameter, the change is applied to the controller immediately.



This model is a real-world implementation of Model predictive control, it is visible that the system is integrated with lead card dynamics perceivable by sensor data and localization within the ego vehicle makes it possible to vary speed in dynamic conditions.

This a fairly moderate model in terms of AV technology and we need to improve it to implementable, which can be achieved my testing in the lab.

MPC optimizes each time step it gives us the least integrated absolute error.

Lane keeping assistance

The Lane Keeping Assist System block simulates a lane keeping assist (LKA) system that keeps an ego vehicle traveling along the centre of a straight or curved road by adjusting the front steering angle. The controller reduces the lateral deviation and relative yaw angle of the ego vehicle with respect to the lane centreline. The block computes optimal control actions while satisfying steering angle constraints using adaptive model predictive control (MPC).

Input:

Road curvature: Road curvature, specified as 1/*R*, where *R* is the radius of the curve in meters.

The road curvature is:

- Positive when the road curves toward the positive Y axis of the global coordinate system.
- Negative when the road curves toward the negative Y axis of the global coordinate system.
- Zero for a straight road.

Ego vehicle velocity: Ego vehicle velocity in m/s.

Ego vehicle lateral deviation: Ego vehicle lateral deviation in meters from the centerline of the lane.

Angle from lane centerline: Ego vehicle longitudinal axis angle in radians from the centerline of the lane.

Minimum front steering angle: Minimum front steering angle constraint in radians. Use this input port when the minimum steering angle varies at run time.

Maximum front steering angle: Maximum front steering angle constraint in radians. Use this input port when the maximum steering angle varies at run time.

Controller optimization enables signal: Controller optimization enable signal. When this signal is:

- Nonzero, the controller performs optimization calculations and generates a **Steering angle** control signal.
- Zero, the controller does not perform optimization calculations. In this case, the **Steering angle** output signal remains at the value it had when the optimization was disabled. The controller continues to update its internal state estimates.

Steering angle applied to ego vehicle: Actual steering angle in radians applied to the ego vehicle. The controller uses this signal to estimate the ego vehicle model states. Use this input port when the control signal applied to the ego vehicle does not match the optimal control signal computed by the model predictive controller. This mismatch can occur when, for example:

- The Lane Keeping Assist System is not the active controller. Maintaining an accurate state estimate when the controller is not active prevents bumps in the control signal when the controller becomes active.
- The steering actuator fails and does not provide the correct control signal to the ego vehicle.

State matrix of ego vehicle predictive model: State matrix of ego vehicle predictive model. The number of rows in the state matrix corresponds to the number of states in the predictive model. This matrix must be square.

The ego vehicle predictive model defined by **Vehicle dynamics matrix A**, **Vehicle dynamics matrix B**, and **Vehicle dynamics matrix C** must be minimal.

Input to state matrix of ego vehicle: Input-to-state matrix of ego vehicle predictive model. The number of rows in this signal must match the number of rows in **Vehicle dynamics matrix A**.

The ego vehicle predictive model defined by **Vehicle dynamics matrix A**, **Vehicle dynamics matrix B**, and **Vehicle dynamics matrix C** must be minimal.

State to angle matrix of ego vehicle: State-to-output matrix of ego vehicle predictive model. The number of columns in this signal must match the number of rows in **Vehicle dynamics matrix A**.

The ego vehicle predictive model defined by **Vehicle dynamics matrix A**, **Vehicle dynamics matrix B**, and **Vehicle dynamics matrix C** must be minimal.

Output:

Front Steering angle control signal: Front steering angle control signal in radians generated by the controller. The front steering angle is the angle of the front tires from the longitudinal axis of the vehicle. The steering angle is positive towards the positive lateral axis of the ego vehicle.

Parameters:

Define ego vehicle model using vehicle properties: Select this parameter to define the ego vehicle model used by the MPC controller by specifying properties of the ego vehicle. The ego vehicle model is the linear model from the front steering angle to the lateral velocity and yaw angle rate.

To define the vehicle model, specify the following block parameters:

- Total mass
- Yaw moment of inertia
- Longitudinal distance from centre of gravity to front tires
- Longitudinal distance from centre of gravity to rear tires
- Cornering stiffness of front tires
- Cornering stiffness of rear tires

Initial state matrix of ego vehicle predictive model: Initial state matrix of ego vehicle predictive model. The number of rows in the state matrix corresponds to the number of states in the predictive model. This matrix must be square.

The initial ego vehicle predictive model defined by **A**, **B**, and **C** must be minimal.

Typically, the ego vehicle model varies over time. To update the state matrix at run time, use the **Vehicle dynamics A** input port.

Input to state matrix of ego vehicle: Initial input-to-state matrix of ego vehicle predictive model. The number of rows in this parameter must match the number of rows in **A**.

The initial ego vehicle predictive model defined by **A**, **B**, and **C** must be minimal.

Typically, the ego vehicle model varies over time. To update the input-to-state matrix at run time, use the **Vehicle dynamics B** input port.

Initial state to angle matrix of ego vehicle: Initial state-to-output matrix of ego vehicle predictive model. The number of columns in this parameter must match the number of rows in **A**.

The initial ego vehicle predictive model defined by **A**, **B**, and **C** must be minimal.

Typically, the ego vehicle model varies over time. To update the state-to-output matrix at run time, use the **Vehicle dynamics C** input port.

Initial velocity of ego vehicle: Initial velocity of the ego vehicle model when the lane-keeping assist is enabled in m/s. This velocity can differ from the actual ego vehicle initial velocity.

Total transport lag in ego vehicle model: Total transport lag, τ, in the ego vehicle model in seconds. This lag includes actuator, sensor, and communication lags. For each input-output channel, the transport lag is approximated by:

$$(1/\tau s+1)$$

Lane Keeping constraints:

Minimum front steering angle: Minimum front steering angle constraint in radians.

If the minimum steering angle varies over time, add the **Minimum steering angle** input port to the block by selecting **Use external source**.

Maximum front steering angle: Maximum front steering angle constraint in radians.

If the maximum steering angle varies over time, add the **Maximum steering angle** input port to the block by selecting **Use external source**.

Model Predictive controller settings

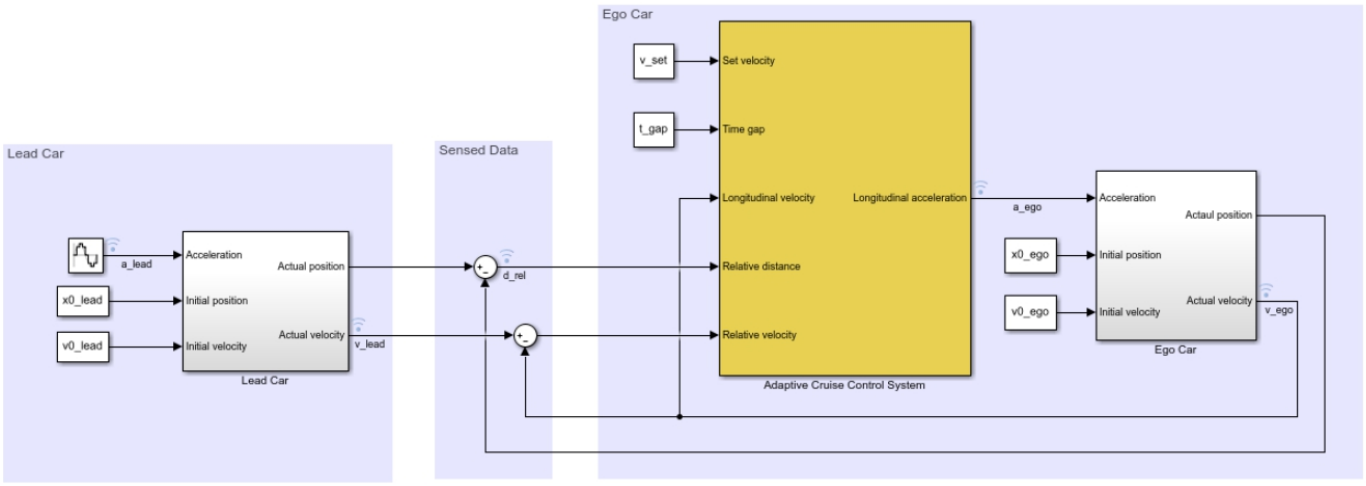
Controller sample time: Controller sample time in seconds.

Controller prediction horizon: Controller prediction horizon steps. The controller prediction time is the product of the sample time and the prediction horizon.

Closed loop controller performance: Closed-loop controller performance. The default parameter value provides a balanced controller design. Specifying a:

- Smaller value produces a more robust controller with smoother control actions.
- Larger value produces a more aggressive controller with a faster response time.

When you modify this parameter, the change is applied to the controller immediately.



SUMMARY

This document stated what can be achieved and how it can be achieved, now the only question is where to do it, and this project has no scope if we cannot do it in the V-CARE facility with the right equipment and precious resources made available by VIT.

We were able to complete this research at home but we cannot further this work anymore, we need physical access to the VIT Campus, the vehicle these protocols will be implementing and real-time troubleshooting.

Said that, we were able to make a working code for the CARLA simulator which can be used for multiple scenarios of 3D reconstructed file of any scenarios.

The MATLAB model for all the stated are control systems, which are very accurate.

REFERENCES

- **An Open Approach to Autonomous Vehicles:** Shinpei Kato; Eijiro Takeuchi Yoshio Ishiguro Yoshiki Ninomiya Kazuya Takeda Tsuyoshi Hamada
- **Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations:** Daniel J.Fagnant Kara Kockelman
- **Planning and Decision-Making for Autonomous Vehicles: Wilko Schwarting,Javier Alonso-Mora, and Daniela Rus**
- **Coursera: Introduction to Self Driving**
- **Coursera: State Estimation and localization**