Autonomous Car using a MPC

Jan-Ruben Schmid

Karol Szurkowski,

Nathan Durocher,

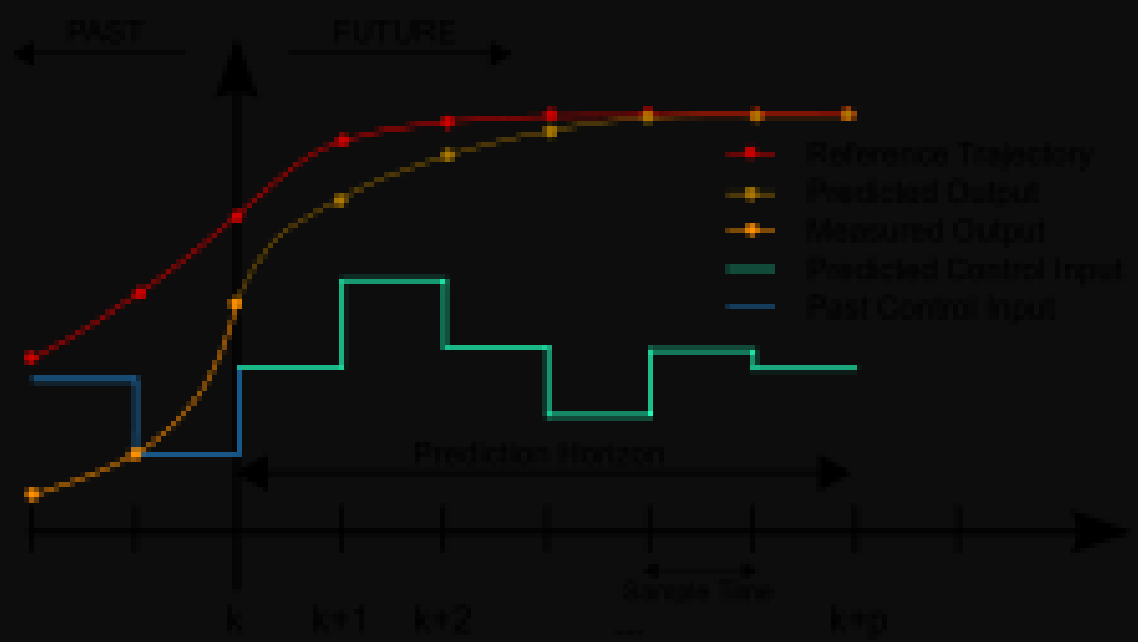
Vincenzo Coppola

Motivation

Where am I?

What is around me?

How to drive?



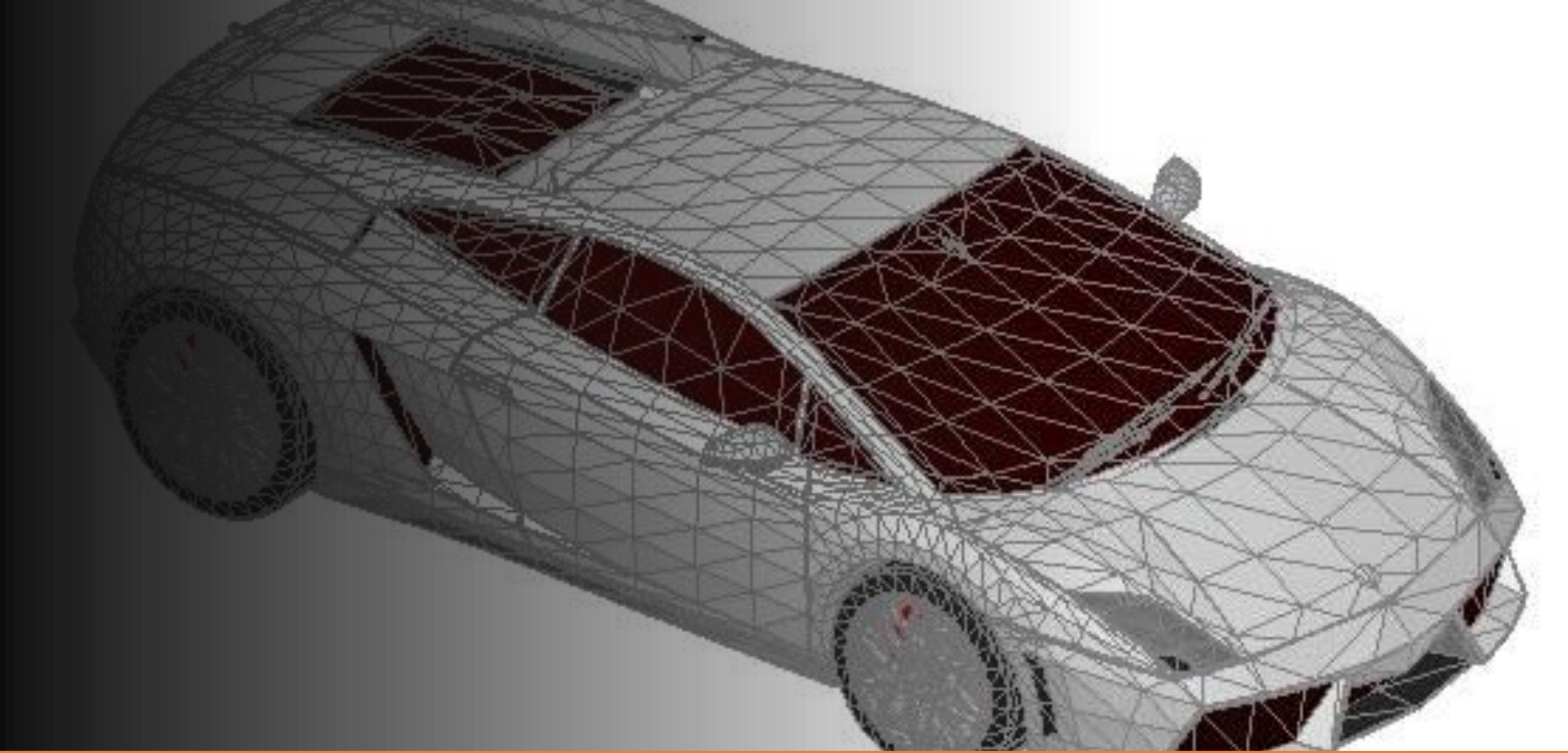
Overview of MPC

- Estimate best driving experience
  - Smoothness
  - Safety
  - Efficiency

  while keeping within physical limits (constraints)

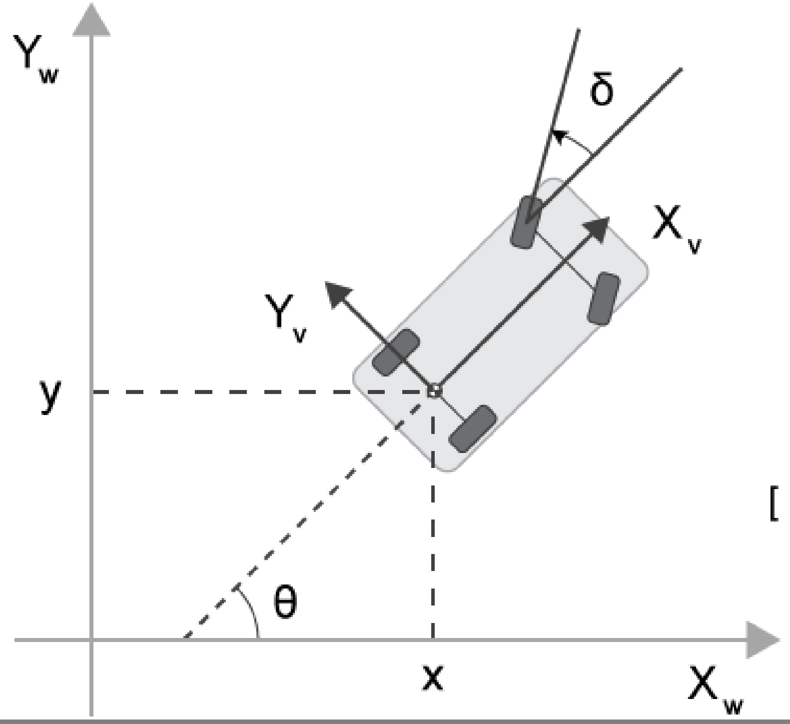
- Get current state from sensors
  - Predict and optimize controls over finite time horizon
  - Take only 1st step

Our approach



1) Car Model
2) Path creation
3) Path following with MPC
4) Simulation in Gazebo

Spatial Parameters



$X_{v,}$ Y_v	Vehicle Coordinate System
$X_{w,}$ Y_w	World Coordinate System
[x, y, θ]	Vehicle Pose
δ	Steering Angle

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & sin\theta \\ -sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

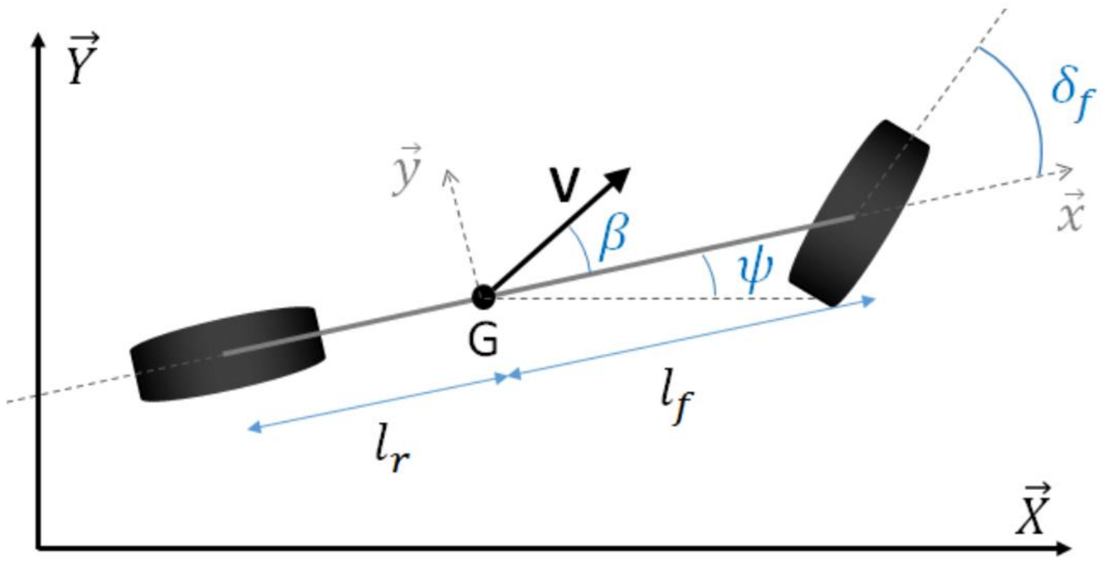
Dynamic Model

$$x_{k+1} = x_k + v_k cos(\psi_k)\Delta t$$

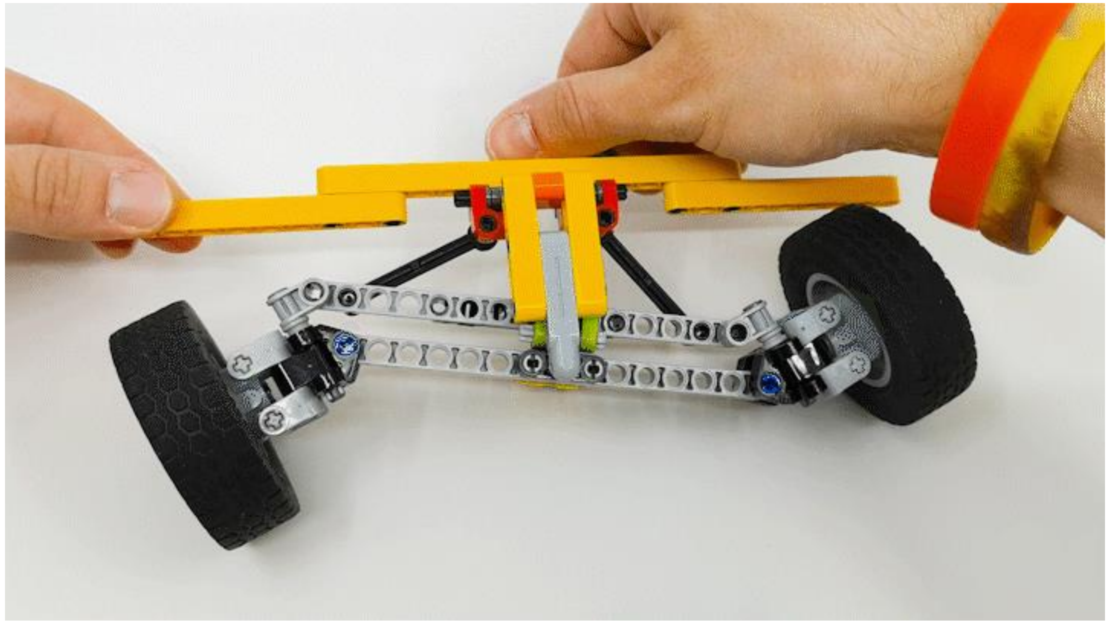
$$y_{k+1} = y_k + v_k sin(\psi_k)\Delta t$$

$$\psi_{k+1} = \psi_k - \frac{v_k}{l_f}\delta_k \Delta t$$

$$v_{k+1} = v_k + a_k \Delta t$$



MPC – Constraints

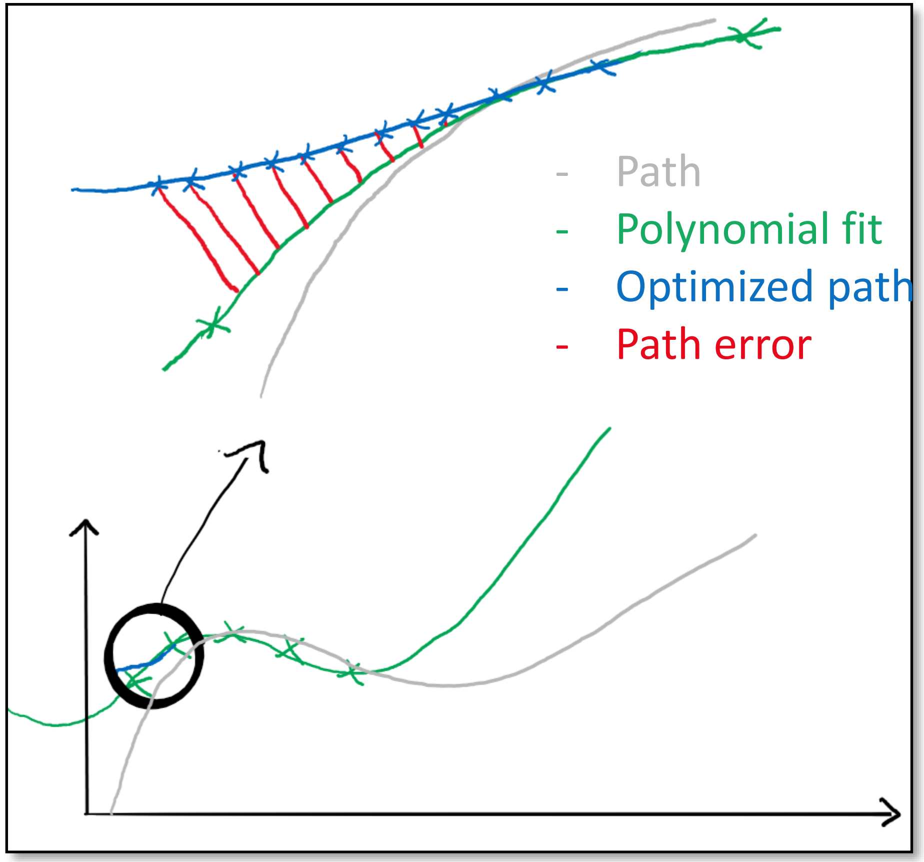


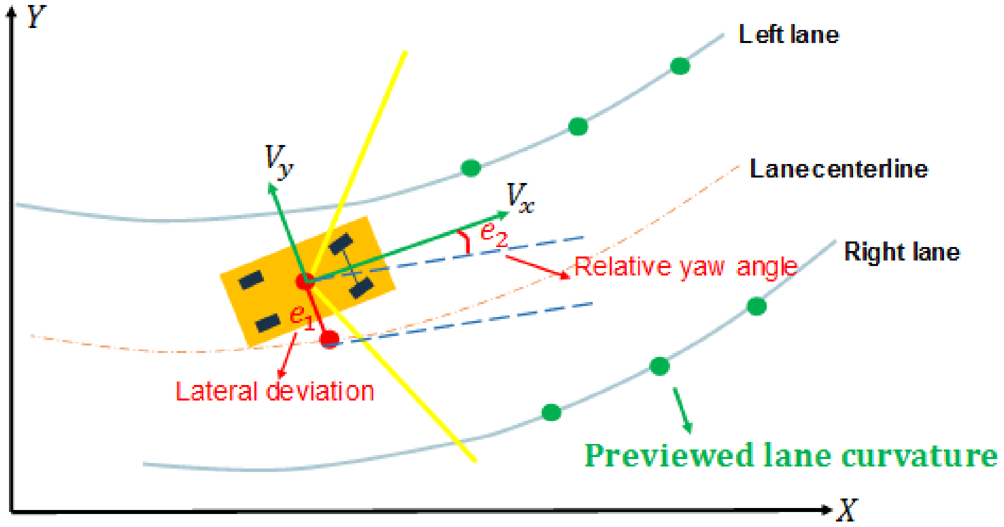
$$-25° \leq \delta \leq 25°$$

$$-3m/s^2 \leq a \leq 3\ m/s^2$$

$$v_{ref} = 11\frac{m}{s}\left(\sim 40\frac{km}{h}\right)$$

MPC – Path planning





$$Pe_{k+1} = f(x_k) - y_k + v_k sin(\psi_k)\Delta t$$

$$He_{k+1} = \psi_k - \psi_{des} + \frac{v_k}{l_f}\delta_k \Delta t$$

MPC – Cost function

$$f(x) = \sum_{t=1}^{N} w_{P_e} \| \; P_e \|^2 + \; w_{H_e} \| \; H_e \|^2 + w_v \| v_t - v_{target} \|^2$$
$$+ \sum_{t=1}^{N-1} w_\delta \| \delta_t \|^2 + w_a \| a_t \|^2$$
$$+ \sum_{t=2}^{N} w_{rate_\delta} \| \delta_t - \delta_{t-1} \|^2 + w_{rate_a} \| a_t - a_{t-1} \|^2$$

Solve non-linear optimization problem

```
%% Weights
Wv = 0.2; %cost of velocity difference from reference velocity
Wa = 1; % cost to acceleration
Wd = 50; % cost to turn
WPe = 600;%cost of path error
WHe = 3;% cost of heading error
Wdr = 400;% cost of change in steering angle
War = 1;% cost of change in acceleration
```

$$\min_x f(x) \text{ such that } \begin{cases} ceq(x) = 0 \\ lb \le x \le ub \end{cases}$$

Track Creation

Motivation

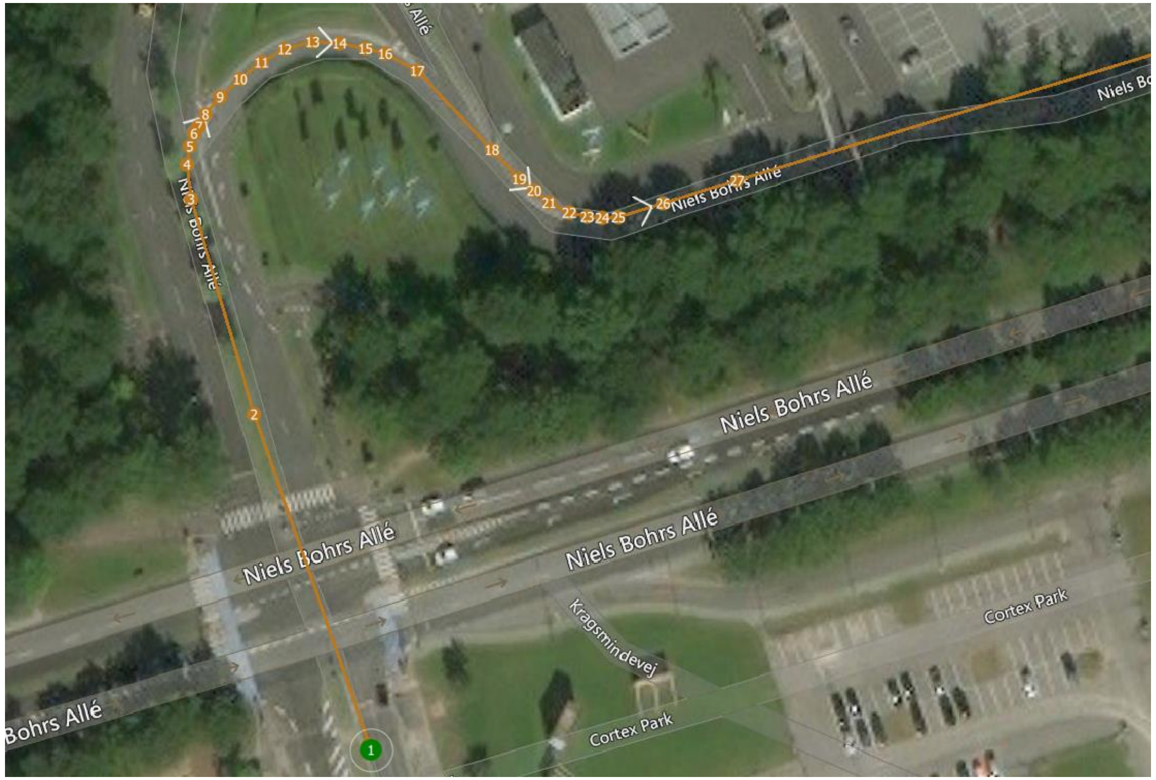
- Reproducible results for MPC
- Create different, comparable paths

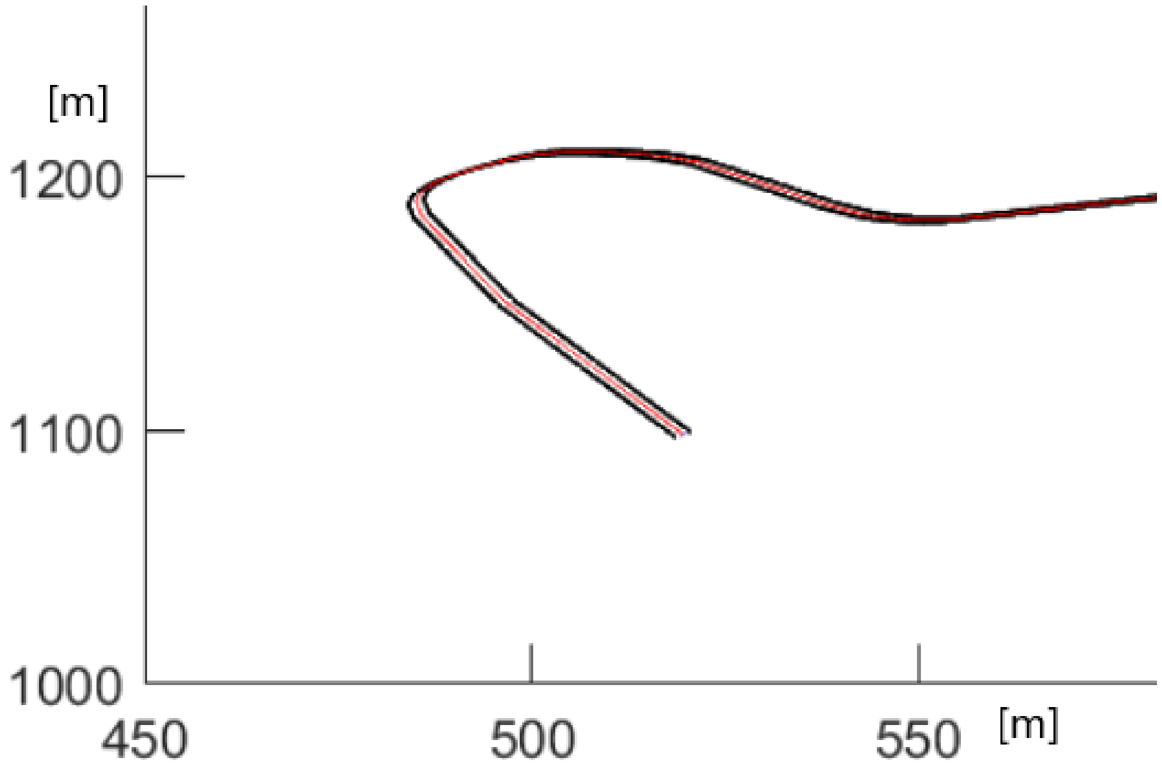
Requirements

- “Smooth” corners and turns
- Fixed length between each waypoint

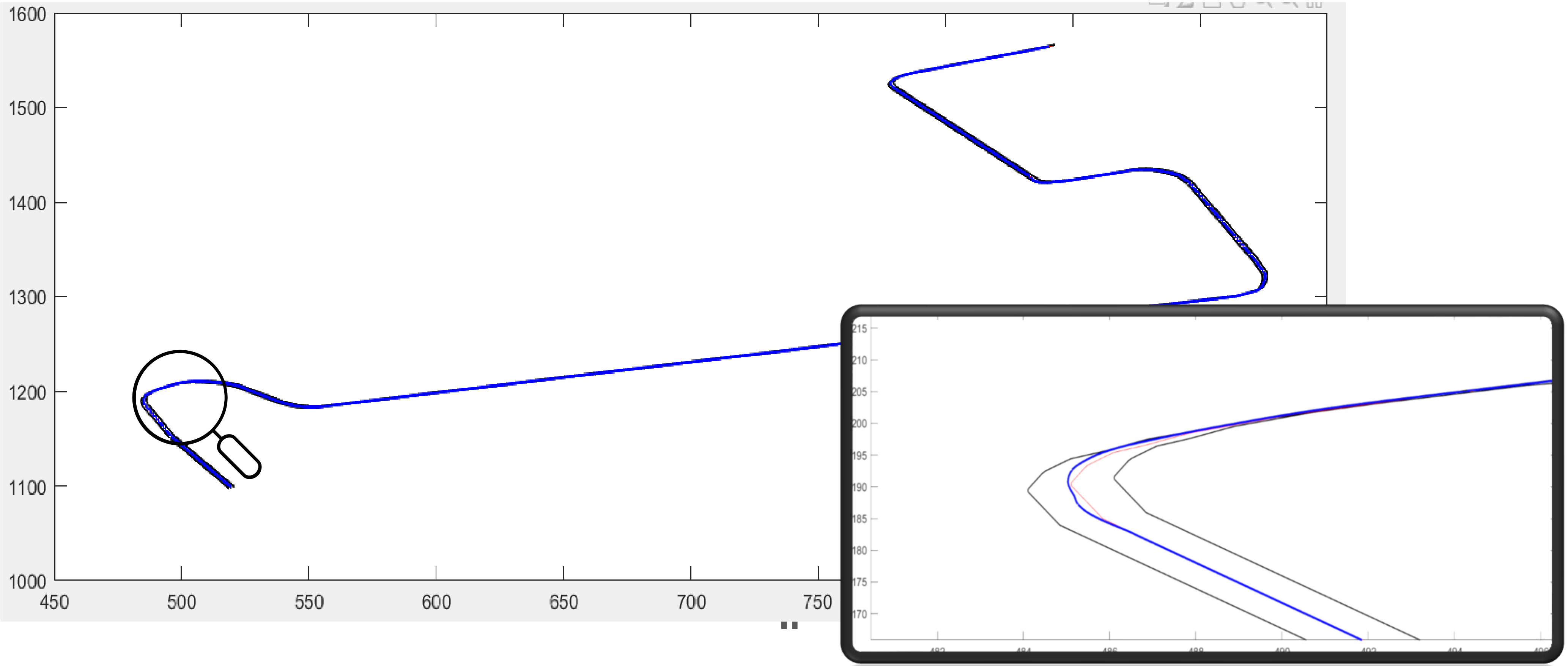
Implementation

- Load path from QGroundControl
- Add intermediate waypoints by linear interpolation
- Choose waypoints with fixed length

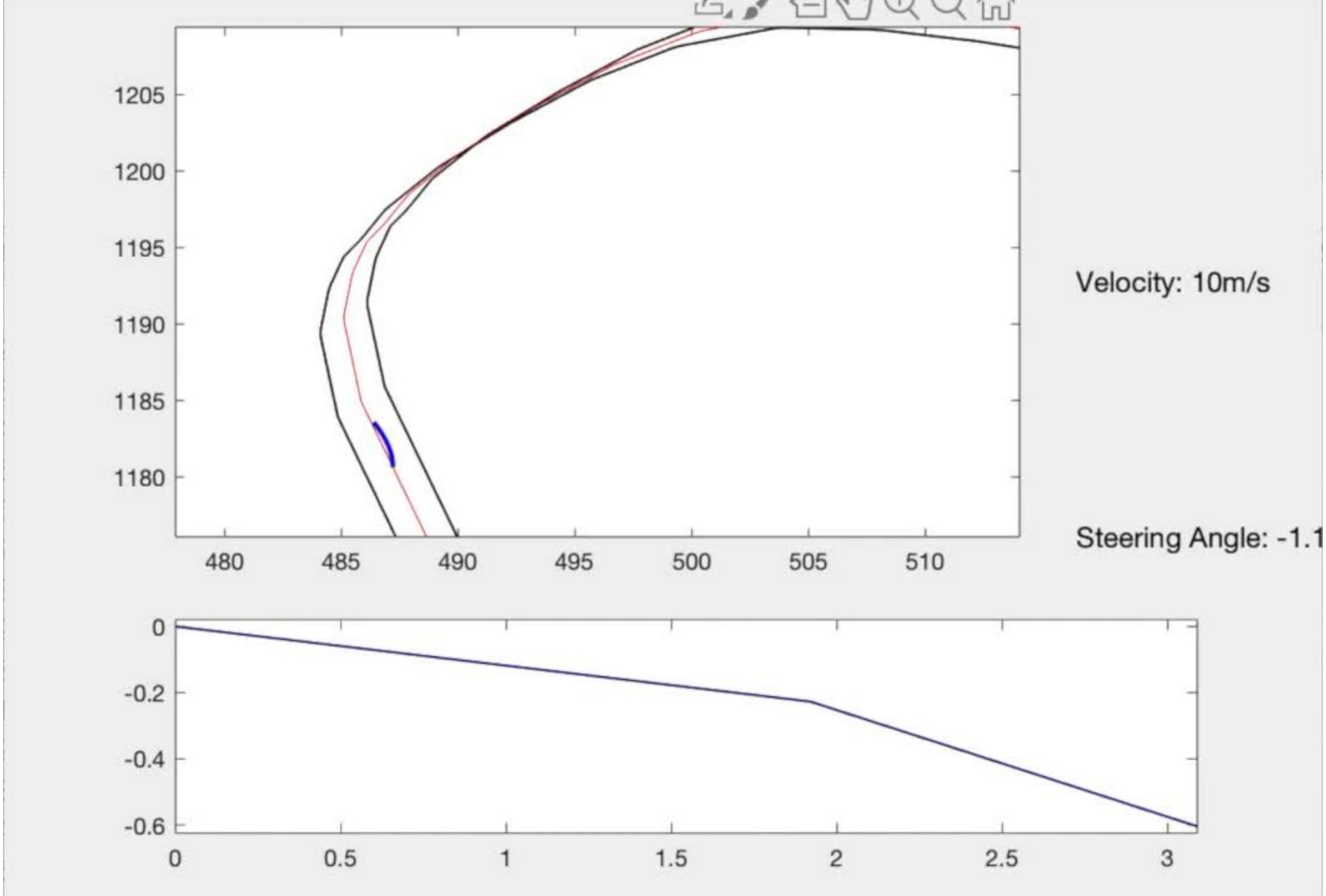


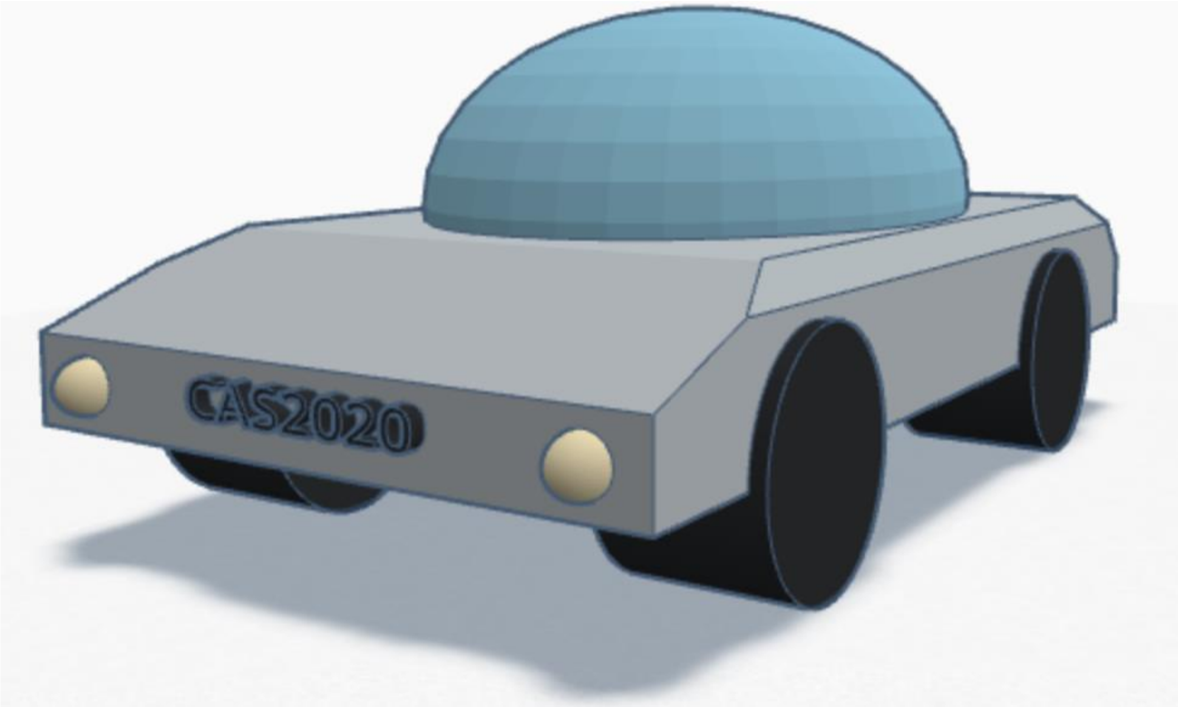


Track Creation


1600
1500
1400
1300
1200
1100
1000
450
500
550
600
650
700
750
215
210
205
200
195
190
185
180
175
170

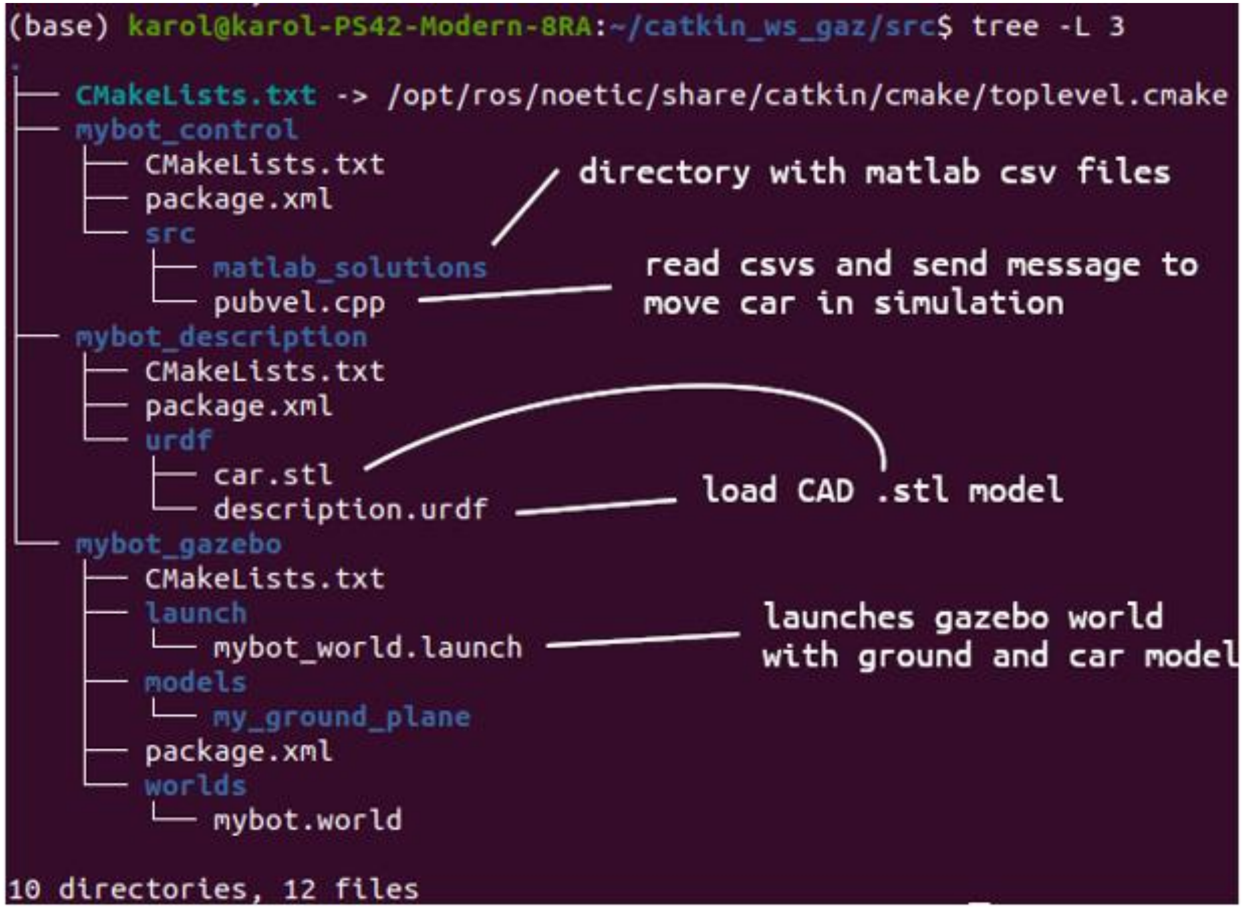
Results


Velocity: 10m/s
Steering Angle: -1.1
1205
1200
1195
1190
1185
1180
480
485
490
495
500
505
510
0
-0.2
-0.4
-0.6
0
0.5
1
1.5
2
2.5
3



Simulation

ROS



`username@hostname:~/catkin_ws_gaz/src$ roscore` - starts master node of ROS

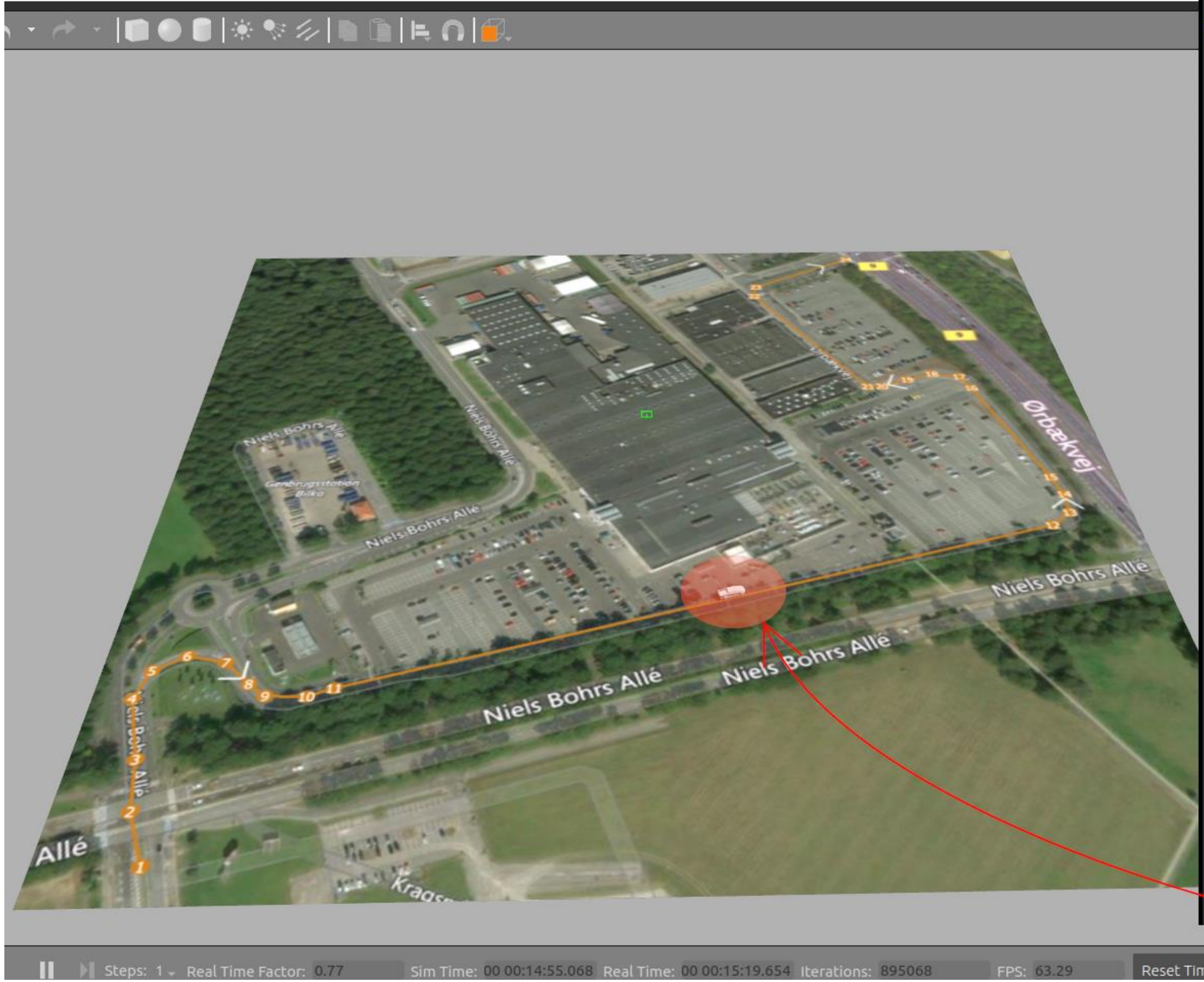
`username@hostname:~/catkin_ws_gaz$ source devel/setup.bash` - adding environment variables to the path to allow ROS to function.

`username@hostname:~/catkin_ws_gaz$ roslaunch mybot_gazebo mybot_world.launch` - launches gazebo service and client with opened world previously defined in an xml-like file

`username@hostname:~/catkin_ws_gaz$ rosrun mybot_control pubvel` - creates new ROS node that sends message with car's position via topic to the Gazebo node.

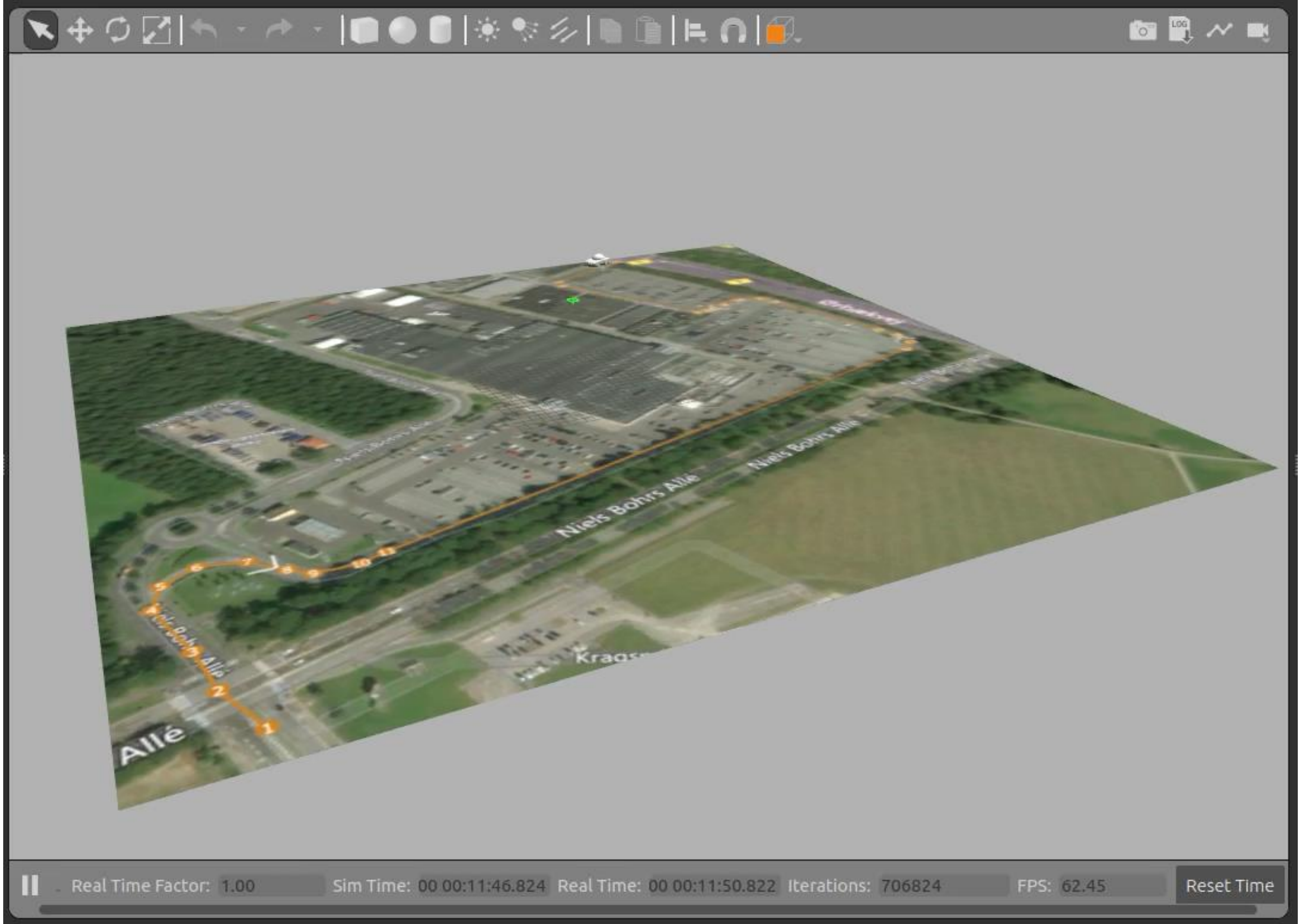
GAZEBO

ROS

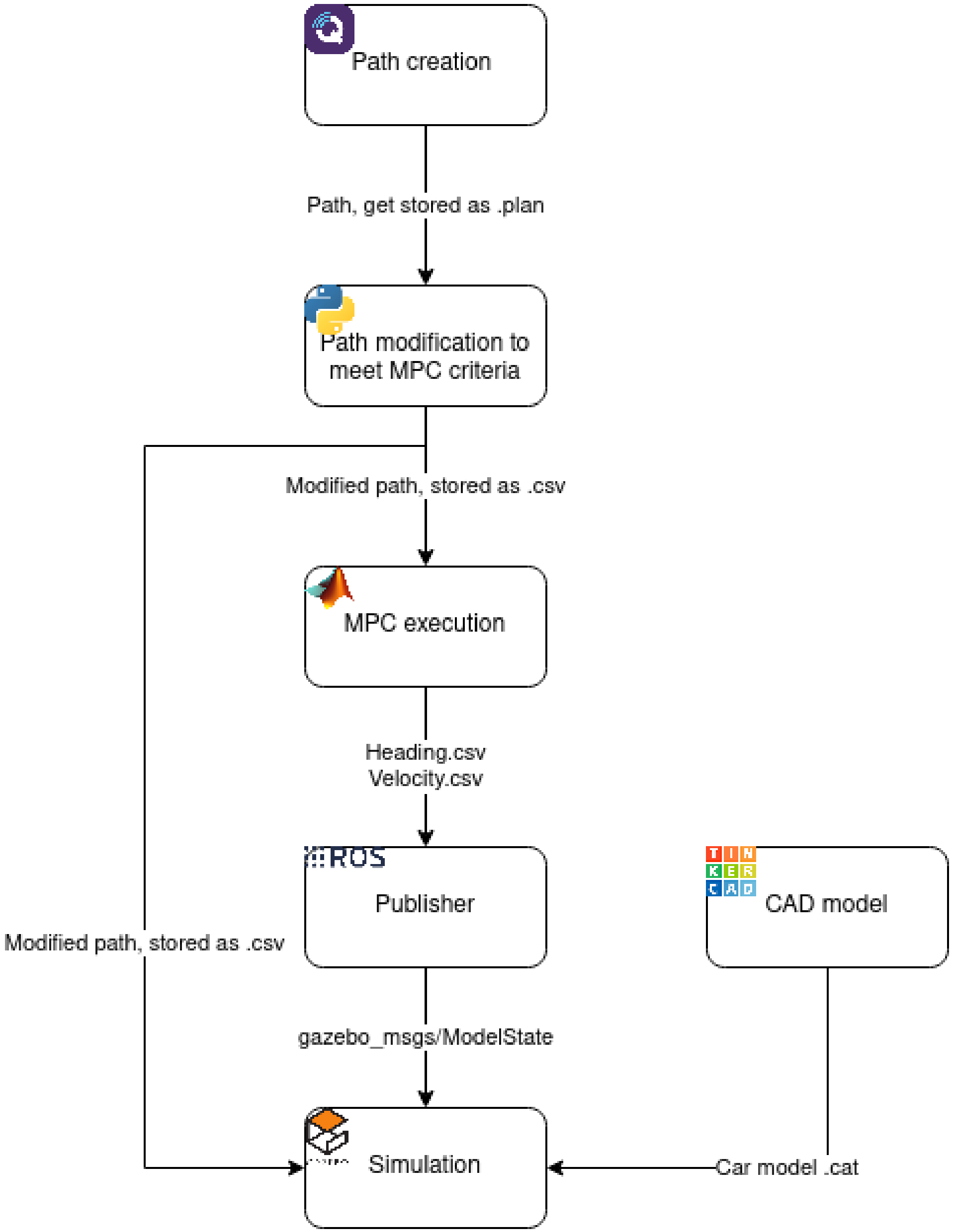


	A	B	C
1	time	orient.z	velocity
2	10	1.8	10
3	20	1.8195	10
4	30	1.8386	10
5	40	1.8576	10
6	50	1.8764	10
7	60	1.8944	10
8	70	1.9115	10
9	80	1.9268	10
10	90	1.9347	10
11	100	1.9506	10

```
    // read csv file
    infilestream.open("src/mybot_control/src/matlab_solutions/Export_
path_time_theta_vel.csv");
    if (infilestream.is_open()){
        std::cout<<"infilestream.is_open()\n";
    }
    // for every row in csv file
    while(infilestream){
        // ommit first row with columns
        if (row_no == 0){
            std::cout<< "count = " << row_no << "| ommit first row of columns\n";
            std::getline(infilestream, row);
            row_no ++;
            continue;
        }
        std::getline(infilestream, row);
        //std::cout<< "\ncount = " << row_no << "| row = " <<row << "\n";

        // calculate new positions given row
        car.update_data(row);

        //send message to gazebo
        geometry_msgs::Pose new_pose;
        new_pose.position.x = car.x_now;
        new_pose.position.y = car.y_now;
        new_pose.position.z = -0.269467;
        new_pose.orientation.x = car.q_rot[0];
        new_pose.orientation.y = car.q_rot[1];
        new_pose.orientation.z = car.q_rot[2];
        new_pose.orientation.w = car.q_rot[3];

        geometry_msgs::Twist new_twist;
        new_twist.linear.x = car.v_x_now;
        new_twist.linear.y = car.v_y_now;
        new_twist.linear.z = 0.0;
        new_twist.angular.x = 0.0;
        new_twist.angular.y = 0.0;
        new_twist.angular.z = 0.0;

        gazebo_msgs::ModelState modelstate;
        modelstate.model_name = "mybot";
        modelstate.reference_frame = "world";
        modelstate.twist = new_twist;
        modelstate.pose = new_pose;

        // Publish the message.
        pub.publish(modelstate);

        // Send a message to rosout with the details.
        ROS_INFO_STREAM("Sending position of mybot:"
        << " msg.position.x =" << modelstate.pose.position.x
        << " msg.position.y =" << modelstate.pose.position.y
```


Niels Bohrs Alle
Niels Bohrs Alle
Allé
Real Time Factor: 1.00
Sim Time: 00 00:11:46.824
Real Time: 00 00:11:50.822
Iterations: 706824
FPS: 62.45
Reset Time

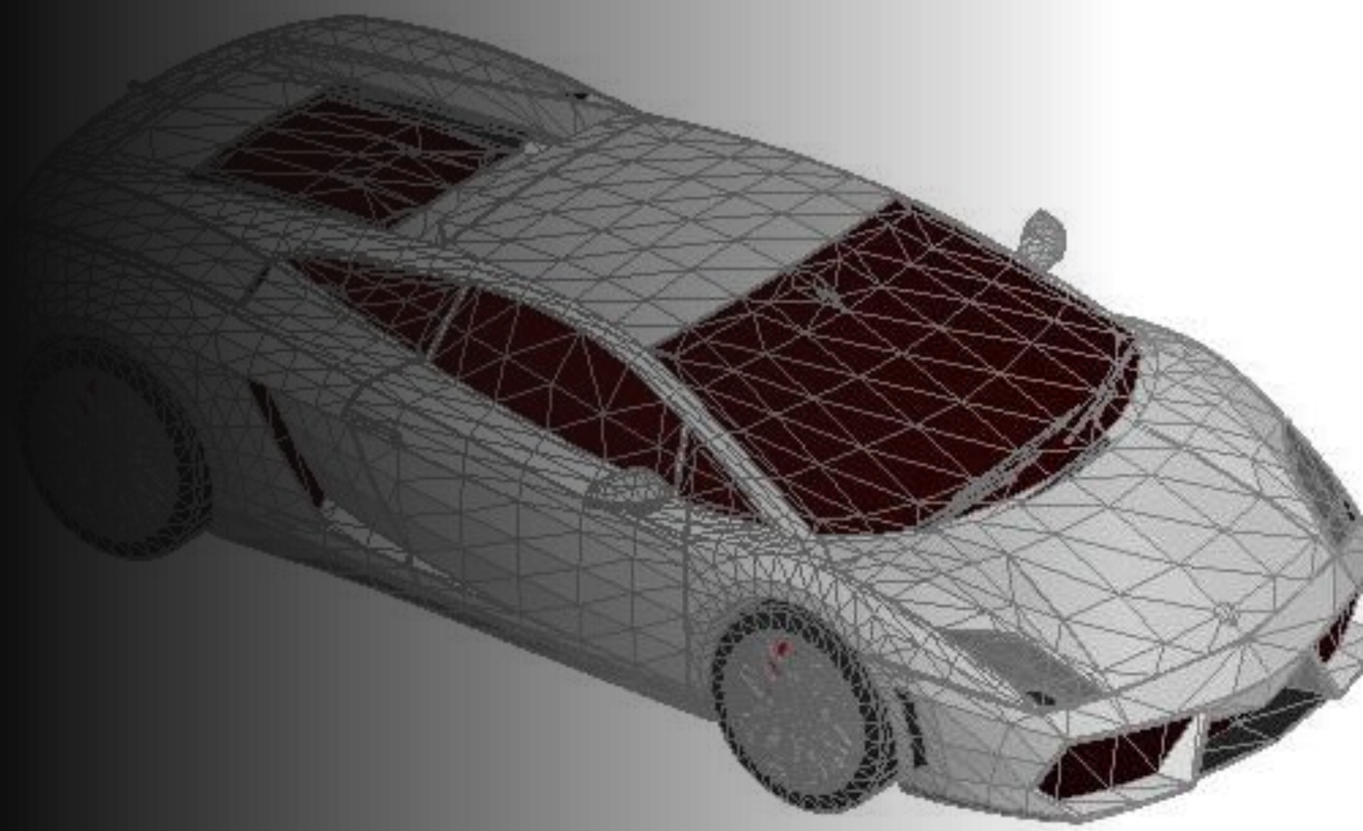
Summary



Path creation
Path, get stored as .plan
Path modification to meet MPC criteria
Modified path, stored as .csv
MPC execution
Heading.csv
Velocity.csv
ROS
Publisher
Modified path, stored as .csv
gazebo_msgs/ModelState
Simulation
CAD model
Car model .cat

Conclusion and Future Work

- Successfully created a MPC
- Further optimize the non-linear solver
- Improve simulation to get closer to real world conditions
- Add obstacle avoidance



Thank you for the attention.