Warsaw University of Technology

FACULTY OF AUTOMOTIVE
AND CONSTRUCTION MACHINERY ENGINEERING

INTERIM THESIS

in the field of study Electric and Hybrid Vehicles Engineering
and specialisation on Autonomous Vehicles

Review of path planning systems

Mert Akdag

Student ID: 309284

Interim thesis supervisor

PhD Eng. Marcin Jasiński

Warsaw 2023

Contents

1. Introduction

1.1 Background

Autonomous vehicles, also known as self-driving cars or driverless cars, are vehicles equipped with advanced technologies that enable them to operate and navigate without human intervention. The general structure of autonomous vehicles encompasses various components that work together to perceive the environment, make decisions, and control the vehicle's motion. In this chapter, I will show the fundamental structure and key components of autonomous vehicles.

1. Perception System:

The perception system is responsible for gathering information about the vehicle's surroundings. It utilizes a combination of sensors, such as cameras, LiDAR (Light Detection and Ranging), radar, and ultrasonic sensors. These sensors provide input for object detection, localization, and mapping, enabling the vehicle to understand its environment. Perception algorithms process the sensor data to detect and track objects, identify road markings, traffic signs, pedestrians, and other vehicles.

2. Localization System:

Localization is the process of determining the precise position and orientation of the vehicle within its environment. Autonomous vehicles use various techniques for localization, including GPS (Global Positioning System), inertial measurement units (IMUs), and odometry. Additionally, sensor fusion algorithms combine the data from multiple sensors to achieve accurate localization, especially in scenarios where GPS signals might be unreliable or unavailable, such as urban environments or tunnels.

3. Mapping System:

Mapping is crucial for autonomous vehicles to have a representation of the environment. High-definition maps are created, which include information about the road geometry, traffic signs, lane markings, and landmarks. These maps are typically prebuilt and stored in the vehicle's memory. The mapping system, along with real-time perception, helps the vehicle understand its position within the mapped environment and aids in path planning and decision-making.

4. Decision-Making System:

The decision-making system processes the information from the perception, mapping, and path planning systems to make decisions about the vehicle's behaviour and actions. It determines how the vehicle should respond to the surrounding environment, including lane changes, stopping at intersections, yielding to pedestrians, or overtaking other vehicles. Decision-making algorithms utilize rule-based logic, machine learning, or probabilistic approaches to generate appropriate responses in real-time.

5. Control System:

The control system is responsible for executing the decisions made by the decision-making system and regulating the vehicle's motion. It converts the desired vehicle behaviour into control commands for acceleration, braking, steering, and other vehicle systems. The control system ensures that the vehicle follows the planned trajectory accurately while maintaining stability and adhering to safety requirements. It may include various control techniques such as PID (Proportional-Integral-Derivative) controllers or model-based controllers.

6. Communication System:

Autonomous vehicles often have a communication system that enables them to exchange information with other vehicles, infrastructure, or a central control center. Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication facilitate cooperative behaviour, such as collision avoidance, traffic coordination, and traffic signal optimization. Communication systems may utilize technologies like Dedicated Short-Range Communication (DSRC) or cellular networks.

7. Redundancy and Safety Systems:

Safety is of paramount importance in autonomous vehicles. Redundancy systems are incorporated to ensure fail-safe operations. Redundant sensors, actuators, and computing systems are deployed to mitigate.

8. Path Planning System:

The path planning system determines the optimal path or trajectory for the vehicle to follow based on the perceived environment, mission objectives, and constraints. Path planning algorithms, such as A* (A-star), Rapidly-exploring Random Trees (RRT), or model predictive control (MPC), generate feasible paths considering factors such as safety, efficiency, and traffic regulations. The path planning system considers obstacles, road geometry, and other vehicles to compute a safe and efficient path for the autonomous vehicle.

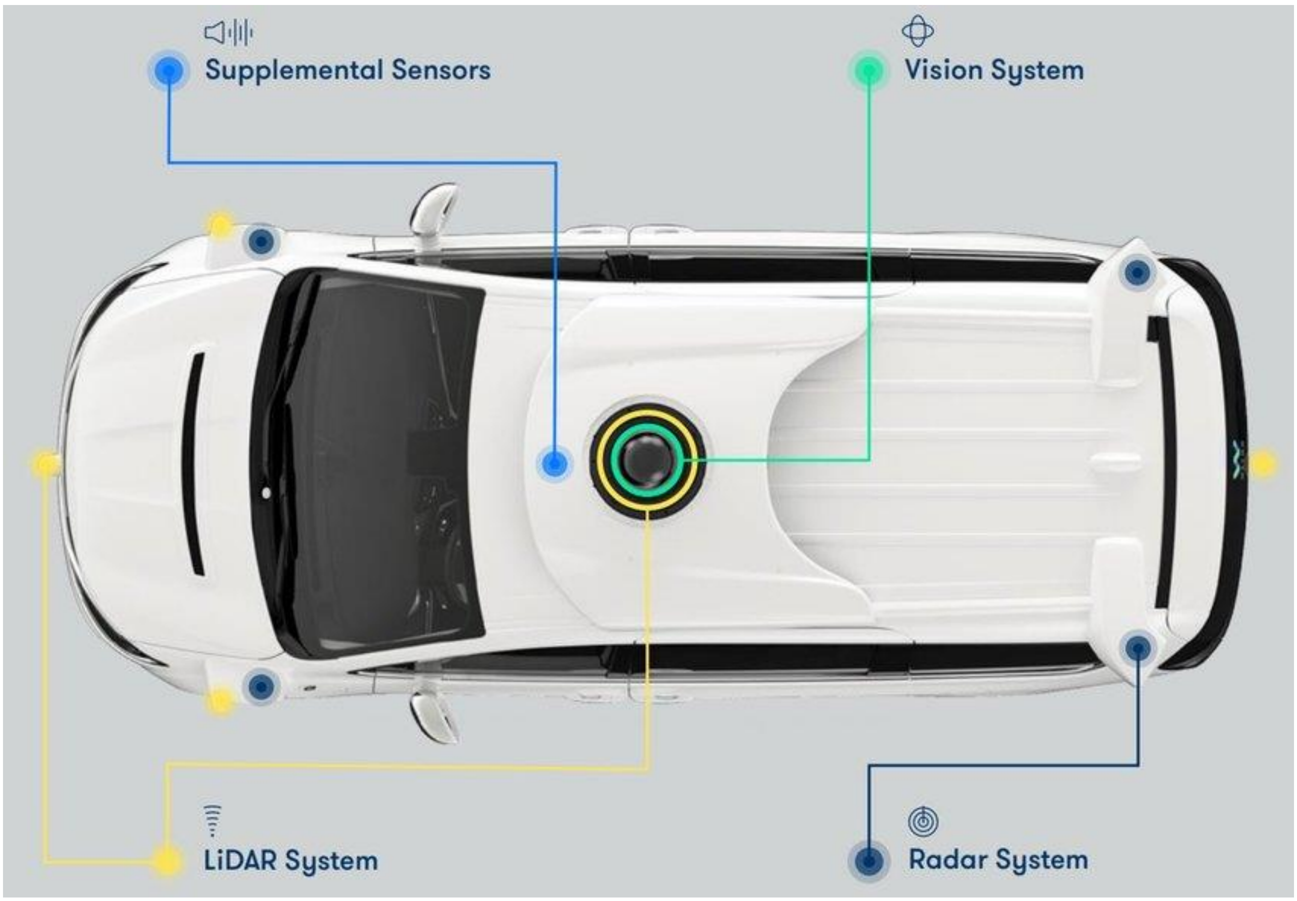


Figure 1: Autonomous vehicles perception sensors[8]

1.2 Autonomous Racing Cars

Autonomous racing cars, also known as self-driving race cars, have emerged as an exciting frontier in the realm of motorsports. Combining cutting-edge technology with the thrill of high-speed competition, these autonomous vehicles are revolutionizing the racing industry. Aim of this thesis to delve into the world of autonomous racing cars, exploring the technology behind them, the challenges they face, and the potential impact on the future of motorsports.

When looking to a historical background of autonomous racing, one of the most well-known and influential events in autonomous vehicle racing is the DARPA (Defense Advanced Research Projects Agency) Grand Challenge, which was first held in 2004. The DARPA Grand Challenge played a crucial role in advancing autonomous vehicle technology, as it spurred significant advancements in perception, decision-making, and control systems. Since then, numerous other racing competitions and leagues have emerged worldwide, such as Roborace where they launched the first autonomous racing series in the world and the first head-to-head race involving autonomous racing vehicles was the Indy Autonomous Challenge, which took place in 2021 and 2022.

Figure 2. The official vehicle of the Indy Autonomous Challenge: Dallara-built AV-21. [9]

2. Path Planning systems for self-driving cars

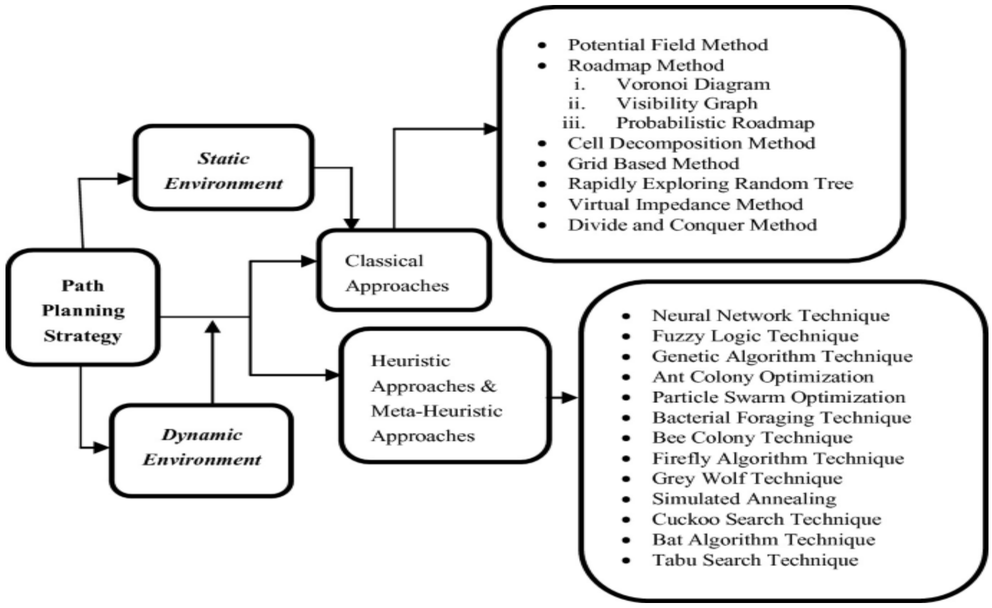


Figure 3. Shows a comprehensive review of path planning algorithms for autonomous vehicles.

The ability of autonomous driving systems to travel across complicated terrain safely and effectively depends on path planning. The path planning problem has been addressed using a variety of techniques and algorithms.

a) Rule-Based Approaches:

Rule-based methods rely on predefined sets of rules and heuristics to generate vehicle trajectories. These rules can be based on traffic regulations, road geometry, and other predefined constraints. For example, a rule-based approach may include rules such as "follow the lane markings," "maintain a safe following distance." While rule-based approaches are relatively simple to implement, they may struggle to handle complex and dynamic traffic situations.

a) Potential Fields:

Potential field methods are based on the concept of treating obstacles as repulsive forces and the vehicle's goal as an attractive force. The vehicle's path is then determined by finding the equilibrium point between these forces. Obstacles generate repulsive forces that push the vehicle away, while the goal generates an attractive force that pulls the vehicle towards it. Potential field methods are intuitive and computationally efficient but can suffer from local minima and oscillations around obstacles.

b) Sampling-Based Approaches:

Sampling-based methods, such as Rapidly-exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM), construct a graph representation of the environment by randomly sampling points and connecting them to form a network of feasible paths. These methods explore the configuration space to generate feasible paths while avoiding obstacles. Sampling-based approaches are effective in complex and dynamic environments, offering scalability and adaptability. However, they may not always guarantee global optimality.

c) Optimization-Based Approaches:

Optimization-based methods formulate the path planning problem as an optimization task. They define an objective function and a set of constraints that capture the desired behaviour of the vehicle. Commonly used optimization techniques include linear programming, quadratic programming, or nonlinear programming. By optimizing the objective function subject to constraints, these methods can generate paths that satisfy specific criteria, such as minimizing travel time or energy consumption. Optimization-based approaches can provide global optimality but may be computationally intensive.

2.1 Optimization-based approaches to path planning

The question is to ask should be: How can a car be steered along a race circuit as swiftly as possible? This calls for an algorithm that can consistently guide the vehicle to follow the planned trajectory in the near future. The objective of path planning for autonomous racing cars is to identify the optimal trajectory, considering various factors such as speed, acceleration, curvature, and the presence of other vehicles. The best racing driver is the one who can navigate a course in the least amount of time with a specific vehicle. The lap time is the only aim as a result. Choosing the trajectory to take is the first step in achieving this goal. In actuality, the ideal trajectory is the optimum trade-off between the shortest track and the track that permits the fastest speeds (track with the minimum curvature).

Optimization-based path planning is an approach that formulates the problem of finding a path for an autonomous vehicle as an optimization task. It aims to find the trajectory that optimizes a specific objective function, while considering various constraints and dynamic limitations. This method has gained popularity in autonomous driving systems as it allows for the incorporation of specific criteria, such as minimizing travel time, energy consumption, or ensuring safety.

Formulation of the Optimization Problem:

In optimization-based path planning, the problem is typically formulated as an optimization task with an objective function and a set of constraints. The objective function captures the desired behaviour or performance criteria of the autonomous vehicle, while the constraints represent limitations on the vehicle's motion, such as maximum acceleration, velocity, or curvature.

Objective Function:

The objective function defines the goal that the optimization algorithm aims to achieve. It can be customized based on the specific requirements of the path planning task. For instance, the objective function might seek to minimize the time taken to reach a destination, maximize the smoothness of the trajectory, or optimize fuel consumption. The choice of the objective function depends on the priorities and constraints of the autonomous vehicle system.

Constraints:

Constraints play a crucial role in optimization-based path planning to ensure that the generated trajectory satisfies specific requirements. These constraints can include physical limitations of the vehicle, such as maximum acceleration, velocity, or curvature. Additionally, constraints related to safety, traffic regulations, and environmental factors might also be considered. By incorporating constraints, the optimization algorithm can generate paths that adhere to the system's limitations and safety requirements.

Optimization Techniques:

Various optimization techniques can be employed in optimization-based path planning, depending on the complexity of the problem and the desired optimality. Linear programming (LP), quadratic programming (QP), or nonlinear programming (NLP) methods are commonly used. These techniques seek to find the optimal solution by iteratively adjusting the trajectory based on the defined objective function and constraints.

Dynamic Constraints and Predictive Models:

To handle dynamic environments and changing conditions, optimization-based path planning often incorporates dynamic constraints and predictive models. Dynamic constraints account for time-varying factors such as the presence of other vehicles, pedestrians, or road conditions. Predictive models allow the optimization algorithm to anticipate the future behaviour of the environment and adjust the trajectory accordingly. This enables the autonomous vehicle to plan ahead and react to potential obstacles or changing traffic scenarios.

Real-Time Adaptation:

Real-time adaptation is an essential aspect of optimization-based path planning for autonomous vehicles. Racing conditions and traffic scenarios can change rapidly, necessitating the ability to adapt the planned trajectory on the fly. Techniques such as model predictive control (MPC) enable iterative optimization, allowing the path planning system to update the trajectory based on real-time sensor inputs and changing constraints. Real-time adaptation ensures that the autonomous vehicle can navigate the environment safely and efficiently.

Advantages of Optimization-Based Path Planning:

1. Ability to customize the objective function based on specific requirements.
2. Consideration of constraints and limitations of the vehicle and the environment.
3. Capability to handle complex and dynamic scenarios.
4. Potential for global optimality, depending on the optimization technique used.
5. Flexibility to incorporate various factors such as safety, energy efficiency, and performance.

Limitations and Challenges:

1. Computational complexity:

Optimization-based path planning can be computationally demanding, especially in real-time applications. Efficient algorithms and approximations are often required to meet real-time constraints.

2. Sensitivity to model inaccuracies:

The performance of optimization-based path planning heavily relies on accurate models and predictions of the environment. Inaccurate models can lead to suboptimal or unsafe trajectories.

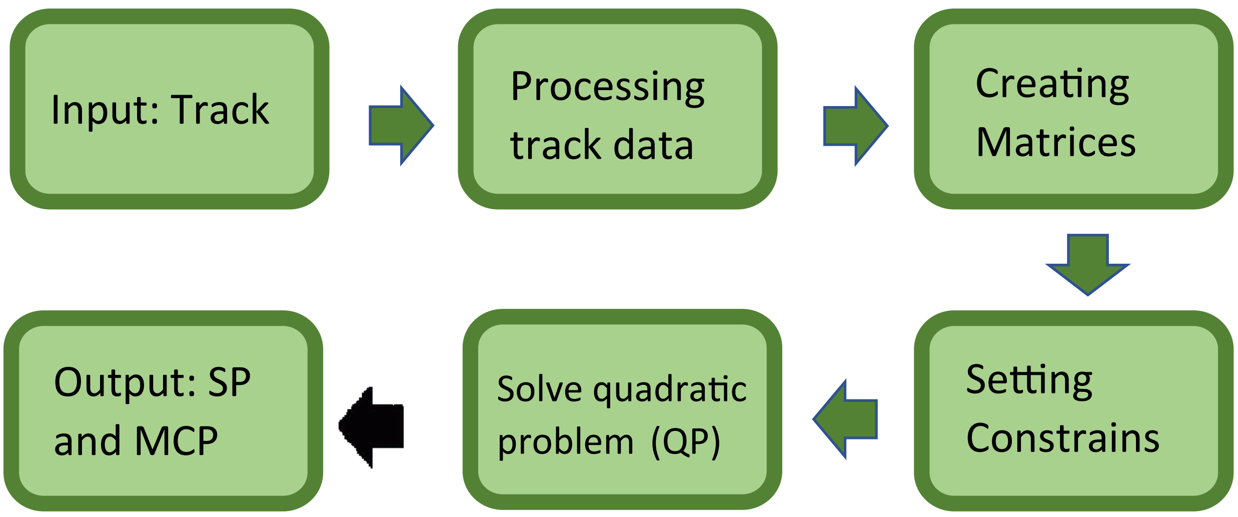
3. High-dimensional search space:

As the complexity of the environment increases, the search space for optimization-based methods can grow exponentially, making it challenging to find optimal solutions.

3. The implementation of optimization problem

To achieve this challenging set of requirements, this thesis uses the mathematical framework of a linearly constrained convex quadratic optimization problem to obtain minimum curvature trajectory.

Workflow of the optimization problem solution:



3.1 Processing Track Data

The main goal of the algorithm is to keep the vehicle in the boundary of inner and outer track limits and complete the lap with highest speed profile which means with minimum curvature trajectory.

- Track data: is imported from outer source in the form of – [x, y, track width to right, track width to left] for this particular implementation of optimization solution, The racetrack chosen for track data is "Autodromo Nazionale Monza".

- x and y coordinates are to denote reference line of the chosen racetrack.

Since the chosen solution must always be inside the track, the algorithm created to find the trajectory with the least curvature solves the constrained minimization problem. The algorithms work by discretizing the track into a number of segments, and at the end of each segment, they identify the location of a specific point on the track using the equation:

$$\begin{aligned}\vec{\mathbf{P}}_i &= x_i\vec{\mathbf{i}} + y_i\vec{\mathbf{j}} \\ &= [x_{r,i} + \alpha_i(x_{l,i} - x_{r,i})]\vec{\mathbf{i}} + [y_{r,i} + \alpha_i(y_{l,i} - y_{r,i})]\vec{\mathbf{j}} \\ &= [x_{r,i} + \alpha_i\Delta x_i]\vec{\mathbf{i}} + [y_{r,i} + \alpha_i\Delta y_i]\vec{\mathbf{j}}\end{aligned}$$

$\vec{P}i$ - position vector for ith point on the track, final output track will be the collection of points;

$\vec{P}$ = [Pi : Pn].

(xi , yi) = x and y coordinates of single segment on track

α_i - a parameter that identifies the position of point ~Pi along the track width. Can be also denoted as a fraction of single segment. When α = 0, Pi is the point of inner track and when α= 1, Pi is the point of outer track.

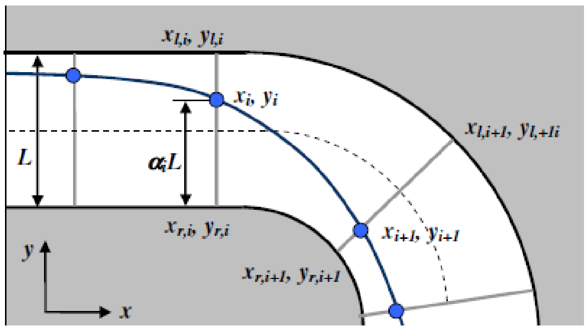


Figure 4. Trajectory discretization.[1]

To obtain an accurate output track, segments must be spaced evenly. For that reason, it must be specified the number of segments that desired to be interpolated from initial track data coordinates.

Then the next step is to offset the reference line (initial x and y coordinates) to obtain inner and outer lines of the track which is possible to set offset variable according to width coordinates provided as [track width to right, track width to left].

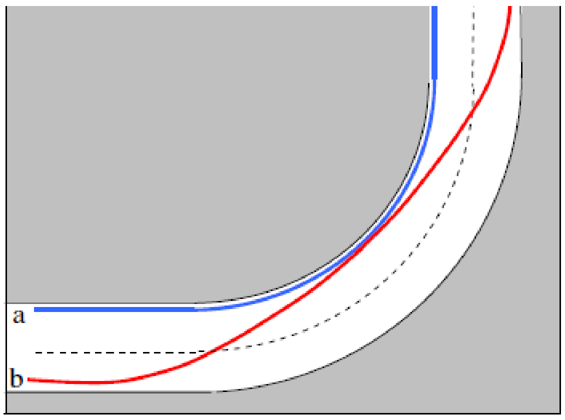


Figure 5. Illustrates the shortest path(a) and the minimum curvature path(b).[1]

3.2 Creation of Matrices

First, difference between outer and inner track coordinates are defined as delta form matrices.

$$\Delta x = x_{out} - x_{in}$$

$$\Delta y = y_{out} - y_{in}$$

Then, quadratic spline method for estimation is used here to get minimum curvature optimization:

$$P_i = a_i + b_i t + c_i t^2$$

To solve quadratic equation, a and b variables are defined as H (square) and B (linear) matrices.

$$t = \frac{s - si}{s_{i+1} - s_{i-1}}$$

All obtained curvatures are summed, in order to find the minimum point:

$$\text{min: } \sum_{i=1}^{n} {c_i}^2$$

As it was mentioned before, it must be defined the constrains where the parameter α_i must be bounded with track limits as:

$$0 \leq \alpha_i \leq 1$$

$$\overline{\alpha} = \{ \alpha_{1,} \alpha_2 \ldots . \alpha_n \}$$

Agreeing to Equation above, the trajectory interior one track segment is represented by a second order polynomial function of t; where t represents the curvilinear abscissa normalized to the length of the ith track portion.

After that, starting and ending points are defined in a same point according to simulate the one completed lap around the track then later on using MATLAB quadratic solver and optimization functions, it was possible to get minimum curvature output track with representing number of iterations where it finally optimized the set of minimum curvature points combined together.

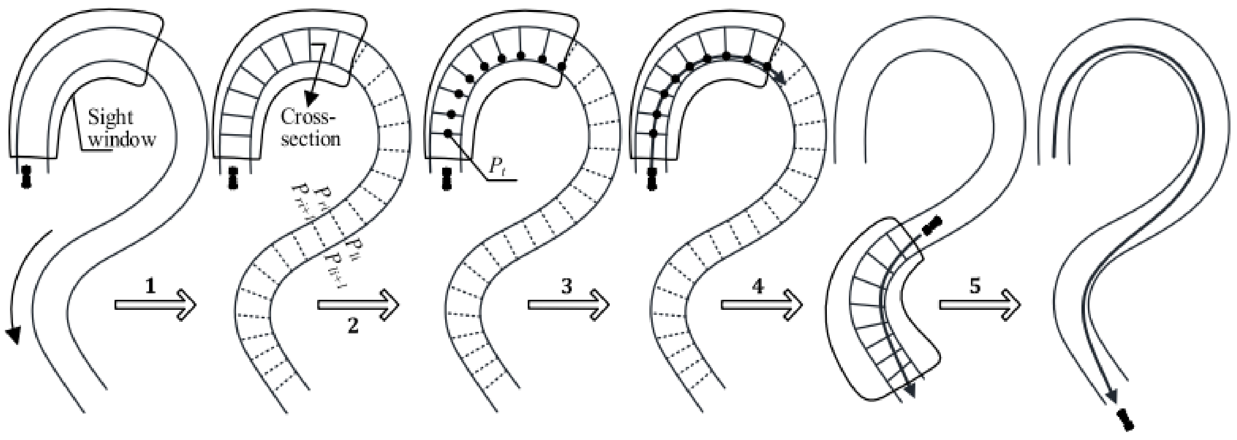


Figure 6. Illustrates the path generation from start to end in steps.[7]

4. Results

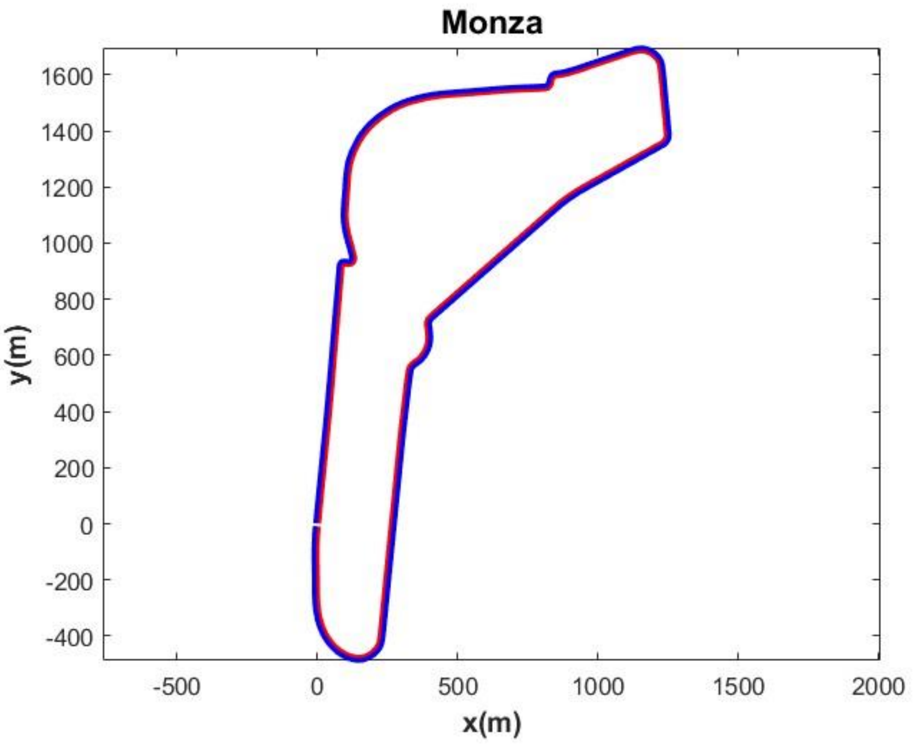


Figure 7. shows the inner outer and reference line of the racetrack.

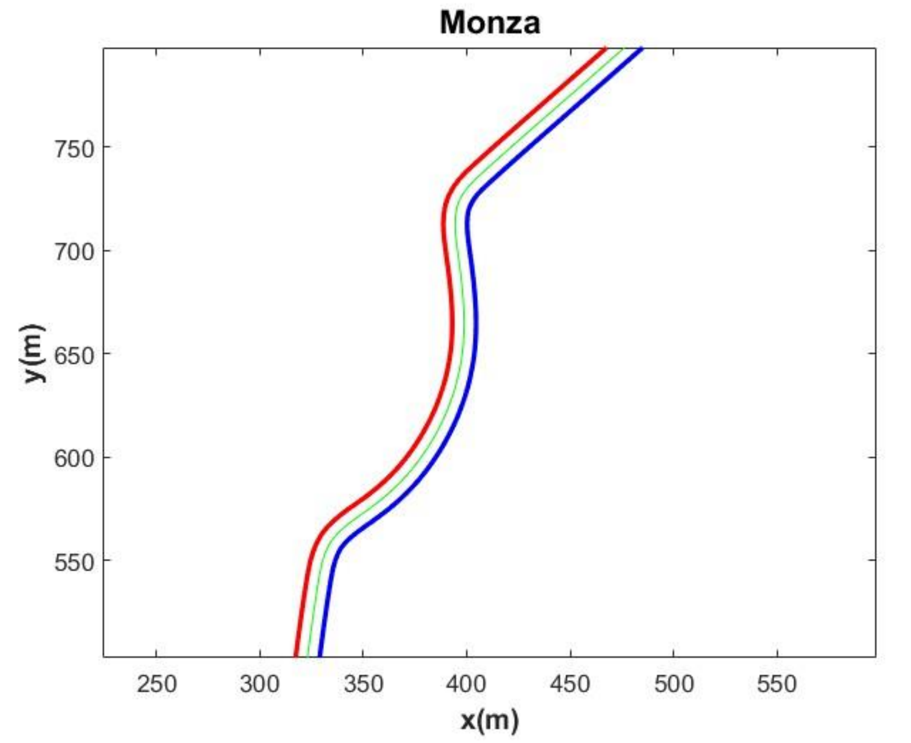


Figure 8. points out the zoomed area of the coordinate system above.

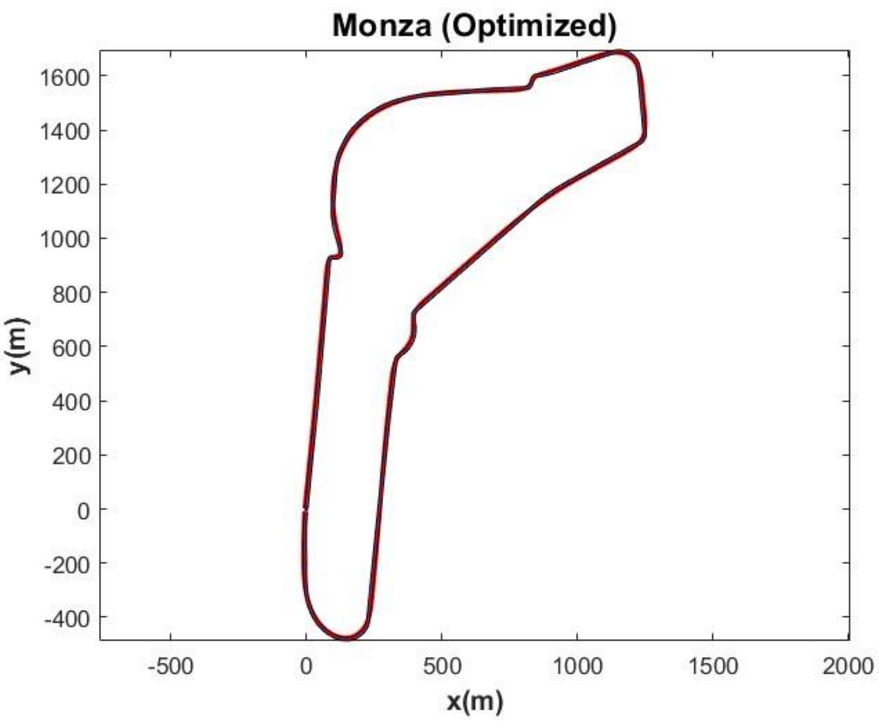


Figure 9. shows the optimized MCP (minimum curvature path) for the racetrack.

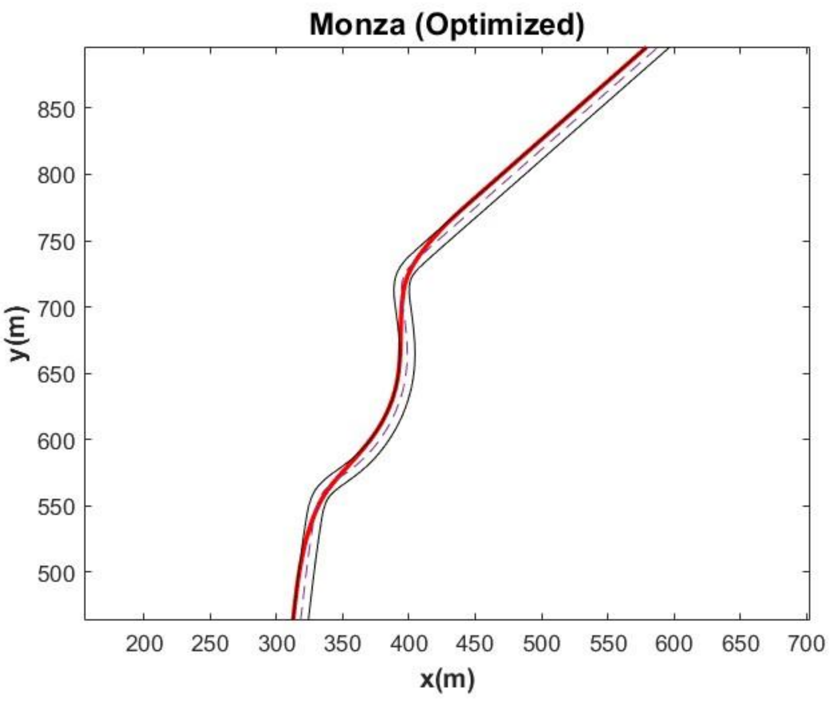


Figure 10. shows the optimized MCP (minimum curvature path) for the racetrack.

5. MATLAB code

```
%% Processing  track data
% track data form = [x y track_width_to_the_right(+ve)
track_width_to_the_left(+ve)]
track = readmatrix("monza_track_coordinates.csv");
name = 'Monza';
% track data
data = track;

% x,y and track width data
x =  data(:,1);
y =  data(:,2);
twr = data(:,3);
twl = data(:,4);

% interpolate data to get finer curve with equal distances between each segment



no_seg = 1500; % higher number of segments causes trajectory to follow the
reference line

pathXY = [x y];
stepLengths = sqrt(sum(diff(pathXY,[],1).^2,2));
stepLengths = [0; stepLengths]; % add the starting point
cumulativeLen = cumsum(stepLengths);
finalStepLocs = linspace(0,cumulativeLen(end), no_seg);
finalPathXY = interp1(cumulativeLen, pathXY, finalStepLocs);

xt = finalPathXY(:,1);
yt = finalPathXY(:,2);
twrt = interp1(cumulativeLen, twr, finalStepLocs,'spline')';
twlt = interp1(cumulativeLen, twl, finalStepLocs,'spline')';

% normal direction for each vertex
dx = gradient(xt);
dy = gradient(yt);
dL = hypot(dx,dy);

% offset curve - anonymous function
xoff = @(a) -a*dy./dL + xt;
yoff = @(a)  a*dx./dL + yt;

% plot reference line
plot(xt,yt,'g')
hold on

% offset data
offset = [-twrt twlt];
for i = 1:numel(xt)
    xin = xoff(offset(i,1));      % get inner offset curve
    yin = yoff(offset(i,1));

    xout  = xoff(offset(i,2));      % get outer offset curve
    yout  = yoff(offset(i,2));
end
```

```
plot(xin,yin,'color','r','linew',2) % plot inner track

plot(xout,yout,'color','b','linew',2) % plot outer track
hold off
axis equal

xlabel('x(m)','fontweight','bold','fontsize',12)
ylabel('y(m)','fontweight','bold','fontsize',12)
title(sprintf(name),'fontsize',14)



%% Matrix Definition

% form delta matrices
delx = xout - xin;
dely = yout - yin;

trackData = [xt yt xin yin xout yout];

n = numel(delx); % number of segments


H = zeros(n);
B = zeros(size(delx)).';




% formation of H matrix (nxn)
for i=2:n-1

    % first row
    H(i-1,i-1) = H(i-1,i-1) + delx(i-1)^2         + dely(i-1)^2;
    H(i-1,i)   = H(i-1,i)   - 2*delx(i-1)*delx(i) - 2*dely(i-1)*dely(i);
    H(i-1,i+1) = H(i-1,i+1) + delx(i-1)*delx(i+1) + dely(i-1)*dely(i+1);

    %second row
    H(i,i-1)   = H(i,i-1)   - 2*delx(i-1)*delx(i) - 2*dely(i-1)*dely(i);
    H(i,i)     = H(i,i )    + 4*delx(i)^2         + 4*dely(i)^2;
    H(i,i+1)   = H(i,i+1)   - 2*delx(i)*delx(i+1) - 2*dely(i)*dely(i+1);

    % third row
    H(i+1,i-1) = H(i+1,i-1) + delx(i-1)*delx(i+1) + dely(i-1)*dely(i+1);
    H(i+1,i)   = H(i+1,i)   - 2*delx(i)*delx(i+1) - 2*dely(i)*dely(i+1);
    H(i+1,i+1) = H(i+1,i+1) + delx(i+1)^2         + dely(i+1)^2;

end

% formation of B matrix (1xn)
for i=2:n-1

    B(1,i-1) = B(1,i-1) + 2*(xin(i+1)+xin(i-1)-2*xin(i))*delx(i-1) +
2*(yin(i+1)+yin(i-1)-2*yin(i))*dely(i-1);
    B(1,i)   = B(1,i)   - 4*(xin(i+1)+xin(i-1)-2*xin(i))*delx(i)   -
4*(yin(i+1)+yin(i-1)-2*yin(i))*dely(i);
    B(1,i+1) = B(1,i+1) + 2*(xin(i+1)+xin(i-1)-2*xin(i))*delx(i+1) +
2*(yin(i+1)+yin(i-1)-2*yin(i))*dely(i+1);
```

```
end

% define constraints
lb = zeros(n,1);
ub = ones(size(lb));

% if start and end points are the same
Aeq = zeros(1,n);
Aeq(1) = 1;
Aeq(end) = -1;
beq = 0;

%% Solver

options = optimoptions('quadprog','Display','iter');
[resMCP,fval,exitflag,output] = quadprog(2*H,B',[],[],Aeq,beq,lb,ub,[],options);

%% Plotting results

% co-ordinates for the resultant curve
xresMCP = zeros(size(xt));
yresMCP = zeros(size(xt));

for i = 1:numel(xt)
    xresMCP(i) = xin(i)+resMCP(i)*delx(i);
    yresMCP(i) = yin(i)+resMCP(i)*dely(i);
end

% plot minimum curvature trajectory
figure
plot(xresMCP,yresMCP,'color','r','linew',2)
hold on

% plot starting line
plot([xin(1) xout(1)], [yin(1) yout(1)],'color','b','linew',2)
plot([xin(2) xout(2)], [yin(2) yout(2)],'color','k','linew',2)

% plot reference line
plot(xt,yt,'--')
hold on

% plot inner track
plot(xin,yin,'color','k')

%plot outer track
plot(xout,yout,'color','k')
hold off
axis equal

xlabel('x(m)','fontweight','bold','fontsize',12)
ylabel('y(m)','fontweight','bold','fontsize',12)
title(sprintf('Monza (Optimized)'),'fontsize',14)

trajMCP = [xresMCP yresMCP];
```

6. Summary

During this project, first the general background information of autonomous vehicles key technologies and general structures were discussed. Afterwards, it was explained that how the autonomous racing cars are invented, and which organizations were taken a role in this new perspective of safer but also, higher controlling and machine learning technology required motorsport. Then in the next chapter, methodologies of different Path Planning Systems were reviewed. With selection of the optimization-based approach for the path planning, in the next chapter, implementation of geometrical trajectory generation for racetrack is explained. During this implementation, optimization solver of MATLAB program is used with the quadratic programming method.

7. Conclusions

It has been explained how to identify the optimal profile and trajectory quickly and accurately. Based on a succession of minimizations, this algorithm that take track geometry into account. It was a successfully computed optimization problem solution by defining minimum of equations and variables to obtain the minimum curvature trajectory profile.

Path planning for autonomous racing cars is a multidisciplinary field that combines mapping, perception, trajectory generation, collision avoidance, optimization, and adaptive learning techniques. By integrating these components effectively, racing cars can navigate racetracks at high speeds while adhering to safety regulations. With advancements in artificial intelligence, machine learning, and sensor technologies, path planning algorithms for autonomous racing cars continue to evolve, promising more efficient and exciting racing experiences in the future.

This approach of finding the best path for the known racetracks, will be useful to train the real racing drivers to complete the race lap with the possible shortest time without any doubt.

References

[1] https://www.sciencedirect.com/science/article/pii/S0045794908000163

[2]https://en.wikipedia.org/wiki/Autonomous_racing

[3]https://github.com/TUMFTM/racetrack-database

[4]https://digitalcommons.du.edu/cgi/viewcontent.cgi?article=2370&context=etd

[5]https://www.mathworks.com/help/matlab/interpolation.html

[6]https://in.mathworks.com/help/optim/ug/quadprog.html

[7]https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8316902

[8]https://www.researchgate.net/publication/337544170_Optimization_Based_Trajectory_Planning_for_Autonomous_Racing

[9]https://www.indyautonomouschallenge.com/racecar