VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

ASSIGNMENT REPORT

NOVEL PATH PLANNING AND TRAJECTORY TRACKING ALGORITHM FOR AUTONOMOUS VEHICLE

Instructor	Dr. Le Xuan Dai	
Group members	Nguyen Quy Hao Nhien	2452911
	Le Tran Minh Phuong	2453025
	Tran Le Tuan Khai	2452517
	Lam Tuan Nguyen	2452846
	Nguyen Minh Quoc Anh	2450001

Ho Chi Minh City, November 2024

Contents

List of Figures

Abstract

The goal of this project is to develop a new approach to solve two well-known problems in robotics engineering: Path Planning and Trajectory Tracking for Automated Vehicles. Our novel approach introduced Artificial Potential Field and Genetic Algorithm to generate a collision-free Bezier spline-based trajectory for the automated vehicle. Then the PID control algorithm is used to help the car steer itself to follow the generated path from the starting point to the ending point.

1 Introduction

1.1 Motivation and Purpose

With the current technological trend, cars, robots, and vehicles have been able to move autonomously basically without any human interaction. Artificial intelligence is such a technology that has provided electric car auto-driving and complex problem solving capabilities to transport humans and objects to desired location. Having seen the vast potential in technological development, we chose Path Planning and Trajectory Tracking problems to start with with the hope of finding new solutions and being a motivational foundation for further researches from other engineers.

(a) Tesla Cybertruck

(b) Vinfast VF9

Figure 1: Electric Cars

1.2 Goal

Currently, AI is the most robust tool to provide the self-driving capability for autonomous vehicles. However, since the technology requires data with high speciality such that it may become hard to obtain (data from specified sensors, high-cost peripherals and devices), we aim to achieve the collision-free motion using another way, which is Genetic Algorithm

2 Project Summary

This project presents an advanced path planning and trajectory tracking system aimed at optimizing navigation in complex environments. The system generates smooth, efficient paths while ensuring safe navigation through obstacle avoidance. It also tracks trajectories with high accuracy, ensuring the system follows the planned path precisely

Path Planning

- **Bézier Curve:** Used to generate smooth and continuous paths, ensuring feasibility in constrained environments.
- **Artificial Potential Fields (APF):** For local obstacle avoidance by generating a recursive danger map that reflects obstacles and their danger level, guiding the system to avoid collisions.
- **Genetic Algoritm (GA):** Inspired by natural selection. The algorithm evaluates multiple possible paths, selects the most efficient ones based on fitness criteria, and evolves the best path through successive generations.

Trajectory Tracking

- **PID Controller:** Ensures accurate following of planned paths, making real-time adjustments to minimize deviations due to environmental disturbances or system dynamics.
- **Normal Distribution:** Introduces noise into the system, closely mimicking real-world conditions and enabling the algorithm to adapt to unpredictable variations in vehicle performance and environmental factors.
- **Binary Search**: Algorithm used to find the projection of a point on the Bezier Curve. It is the key algorithm to support the PID Controller, helping the car stays on track along the path. It also helps the Genetic model validate the feasibility of the Bezier path.

Tools and Libraries

- Python programming language
- numpy library
- Pygame library
- Matplotlib library
- Object-oriented Programming

3 Theoretical Basis

3.1 Bézier Curve

Bezier curves is a set of locally generated points that can be constructed and positioned from a set of control points. The more local control points being constructed, the smoother curve becomes. Bezier curves are widely used in real life for tasks requiring smooth and precise shapes. They are essential in graphic design and digital art for creating scalable vector graphics and smooth font outlines.

3.1.1 Linear Interpolation

By definition of the Bezier Curve, Linear Interpolation is the main foundation of constructing the points on the curve. This is a method of estimating an intermediate value between two points P_0 and P_1, based on a parameter t that runs on the interval $[0; 1]$.

$$P(t) = (1 - t)P_0 + tP_1$$

where t=0 gives P_0 and t=1 gives P_1.

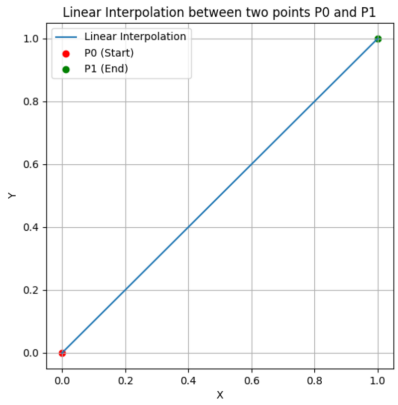


Figure 2: Linear Interpolation

3.1.2 General Form of Bézier Curves

In order to construct a curve, multiple Linear Interpolations can be performed nested inside each other. For example, interpolating P_3 between P_0 and P_1, P_4 between P_1 and P_2 and then do the same with P_5 between P_3 and P_4 will give us a quadratic Bezier Curve. In this case, P_0, P_1 and P_2 are called "control points". The general form of a Bézier curve allows it to be extended to any degree n based on the number of control points $P_0, P_1, \ldots, P_n$. The curve is defined as:

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} (1 - t)^{n-i} t^i P_i$$

Where:

- $B(t)$ is the point on the Bézier curve at parameter $t \in [0, 1]$,
- $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ is the binomial coefficient,
- P_i is the i^{th} control point,
- n is the degree of the Bézier curve, $(n + 1)$ is the number of control points.

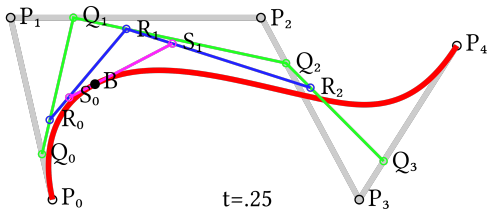
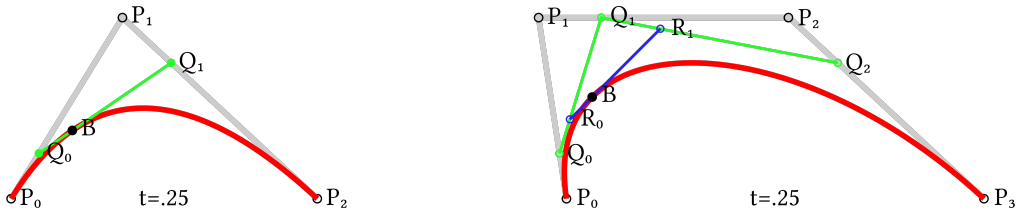


Figure 3: General Form of Bézier Curves

Example

- $n = 1$, the curve is a simple straight line (single Linear Interpolation).
- $n = 2$, the curve is a Quadratic Bezier Curve.
- $n = 3$, the curve is a Cubic Bezier Curve.



(a) Quadratic Bézier Curve (b) Cubic Bézier Curve

Figure 4: Some cases of Bézier Curve

3.1.3 Properties

- **Control Points:** The curve starts at P_0 and ends at P_n. Intermediate control points $(P_1, P_2, \ldots, P_{n-1})$ influence the shape but are not necessarily on the curve.
- **Continuity:** Bézier curves are continuous and smooth, making them ideal for graphics and design applications.
- **Scalability:** Adding more control points increases the degree of the curve, allowing for more complex shapes.

3.2 PID Control

PID is a closed loop algorithm use to help a system achieve a pre-determined setpoint. For example, PID could be used to adjust the valve of a water pipe to control the water flow inside fixed at a setpoint. The main formula of PID algorithm is:

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

where

- $u(t)$ is the variable that the system is trying to control
- $e(t)$ is the difference between the current state of the system with the setpoint
- K_P, K_I, K_D are Proportional Gain, Integral Gain and Derivative Gain, these are system's constants needed to be tuned through multiple experiments

3.2.1 Proportional Control (P)

The proportional term in the general formula adjusts the control signal based on the error. By adjusting the Proportional Gain (K_P) high, the system would response more quickly and aggressively to the setpoint, but it may overshoot and oscillate if the value is too high. Setting a low gain value will make the system less responsive than needed.

$$P = K_P \cdot e(t)$$

Here are the results of too low, ideal and too high P-Gain settings:

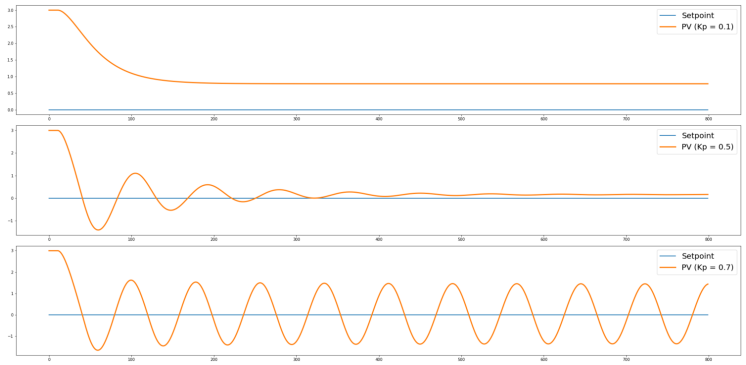


Figure 5: Examples of different P Gain values

However, only the Proportional Control alone is not enough since it does not eliminate the steady-state error (persistent offset) and also the time-consuming convergence to the setpoint.

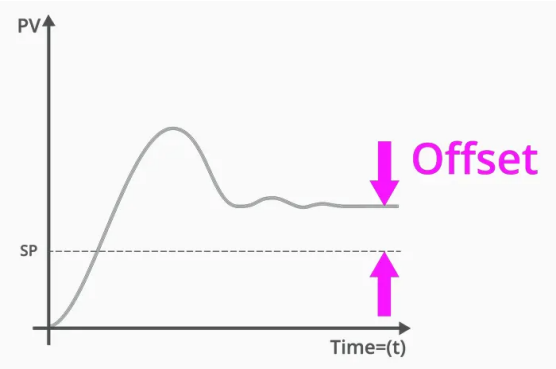


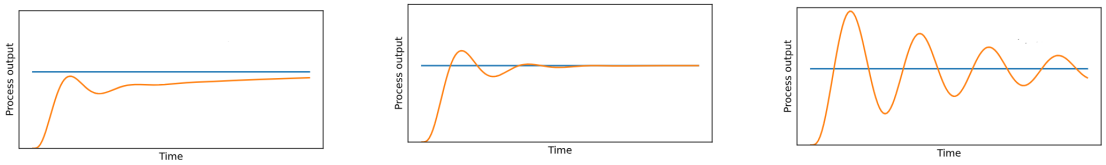
Figure 6: Steady-state Error

3.2.2 Integral Control (I)

To get rid of the steady-state error of the slow P-Controller, the integral term could be utilized to accumulate the error over time and therefore adjusts the control variable to the setpoint:

$$I = K_I \cdot \int_0^t e(t)dt$$

The incorrectly tuned Integral gain may results in under-damped (slow reaction) system or oscillating behavior. The following image contains some examples of the behavior of the system if the Integral Gain is set too low, ideal and too high.



(a) I Gain too low (b) I Gain ideal (c) I Gain too high

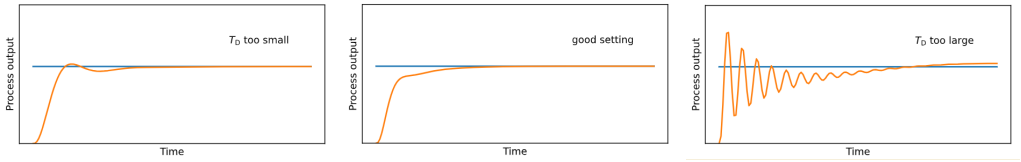
Figure 7: Examples of several values of I-Gain

3.2.3 Derivative Control (D)

The derivative term predicts future errors by considering the rate of error change.

$$D = K_D \cdot \frac{de(t)}{dt}$$

Implementing this term will help increase the system damping effect since the rate of change (velocity) of the error is considered. However, to high Derivative Gain may also lead to extensive overshoot of the system and noise is also a significant factor to be considered when tuning this constant.



(a) D Gain too low (b) D Gain ideal (c) D Gain too high

Figure 8: Examples of several values of D-Gain

3.2.4 Combined PID Control

$$u(t) = K_P e(t) + K_I \int e(t)\,dt + K_D \frac{de(t)}{dt}$$

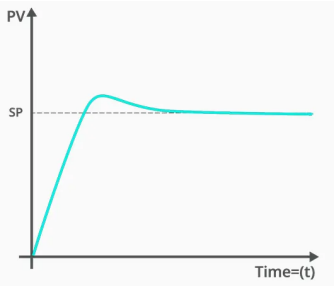


Figure 9: Combination of Proportional, Integral and Derivative controls

In order to make the system behave robustly and perfectly, PID-tuning procedure must be conducted, in which of Proportional, Integral and Derivative Gains must be determined. Each parameter has different physical influence on the system, therefore it will be time-consumming to find the perfect conbination.

3.3 Artificial Potential Field

Artificial Potential Field is an approach to robot Path Planning problem that uses imaginary field generated on the robot's working environment, being a foundation for external Path Planning algorithms. The APF often consists of two types of force:

- Attractive Force centered at the final desired position in robot's navigation acts as a directing tool.
- Repulsive Force placed along the edge of the obstacles helps repelling the robot when it gets close to the obstacle, providing the collision-free capability for the robot's navigation.

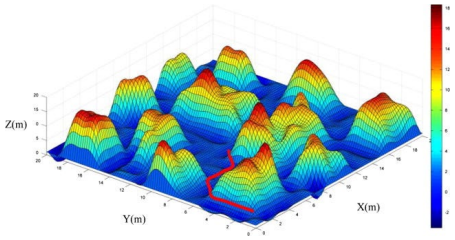
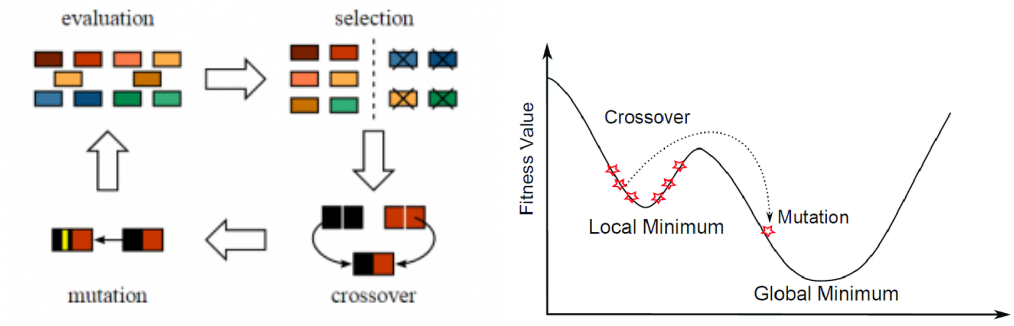


Figure 10: Artificial Potential Field

Another intuitive representation of the APF is by considering about the geographical meaning of it. Each of the obstacles can be assigned a "height", contributing to the total geographical terrain of the robot's working environment. The closer the vehicles gets to an obstacle, the steeper the slope it has to climb, the artificial gravitity will pull the vehicle down, hence, repelling it away from the mountain, or in this case, the obstacle. Through out this project, we will be using the geographic interpretation of APF as a supporting tool for the Genetic Algorithm introduced.

3.4 Genetic Algorithm

Genetic Algorithm is a heuristic adaptive search algorithm, part of the Evolutionary Algorithm which taken inspiration from the natural evolution of organisms. Genetic Algorithms iteratively modiy the genetic encoded chromosome of each individuals in a population with the principle of crossover and mutation, mimicking the evolution. The population then runs through a fitness function to be evaluated, or eliminated if its performance is too terrible.



(a) Genetic Algorithm example (b) Purpose of mutation

Figure 11: Some cases of Bézier Curve

Relying on the natural selection principle, the chromosomes that yielded better performance (elites) will proceed to the next evolution, converging to an answer of the problem that we are trying to solve. Also, the randomness of natural mutation would provided the diversity of individuals that may produce better results, hence helping the model escape out of the local minima to find a better one.

3.5 Gaussian Distribution

The Gaussian distribution, is a cornerstone of probability and statistics, distinguished by its symmetric, bell-shaped curve. Its prevalence in various natural process, which asserts that the aggregate of numerous independent random variables. This help the system to deal with the unexpectable real-life conditions.

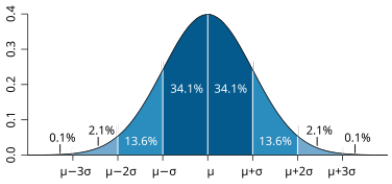


Figure 12: Gaussian Distribution

3.6 Binary Search Algorithm

Binary Search Algorithm is a method of finding an element in a sorted array. It does this by iteratively divide the array in half, looking for the element in either left and right partitions and ignore one them if the element lies on the other. This is a very efficient algorithm in searching problems, especially for those with sorted values like ours because by repeatedly cutting the problem in half, the complexity is on the logarithmic scale.

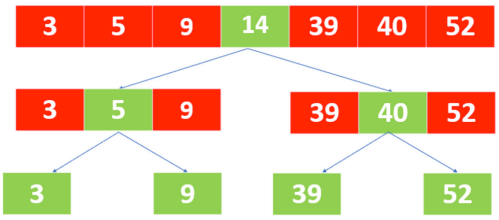


Figure 13: Binary Search Visualization

4 Methodology

Problem Formulation
Given a car in a 2D map with predetermined start point and end point and circular obstacles that have randomly generated radius. The goal of the project is to introduce a collision-free Path Planning Algorithm and a Trajectory Tracking method to guide the car along the path. The factor that we are trying to minimize is the danger of the Bezier path. The closer the path is relative to the obstacles, the more danger the path yields.

4.1 Path Planning Algorithm

The Path Planning Algorithm involves the presence of Artificial Potential Field, Bezier Curve and most importantly, Genetic Algorithm. The general training procedure of our model can be described as follows:

1. **Introduce a Bezier to Chromosome encoding strategy:** The encoding scheme is a universal definition of the chromosome and any latter functionality, model training procedure will have to surround this definition.
2. **Determine a general definition of path's danger:** Artificial Potential Field will present in this part, help generating tools needed to evaluate the chromosomes latter on.
3. **Initialize the first population**: Create random initial chromosomes.
4. **Artificial selection**: Artificial Fitness Function is defined and will be complimented with APF to evaluate the population.
5. **Artificial evolution**: The population will undergo the crossover and mutation, mimicking the natural evolution of animals and organisms.
6. **Loop**: Return to step 4 until the number ofatraining epoch runs out.

4.1.1 Bezier to Chromosome Encoding Strategy

The chromosomes of the Genetic model is defined as a sequence of control points of the Bezier Curve which will act as genes. For instance, a curve that has P_0, P_1, P_2 and P_3 as control points will be encoded into $\mathbf{P} = \{P_0, P_1, P_2, P_3\}$ as chromosome. Since the start and end positions of the car on the 2D map are known, we will assign the first and last control points in the sequence corespond with them, which means the sequence $\{P_0, P_1, P_2, P_3\}$ will become $\{P_S, P_1, P_2, P_E\}$.

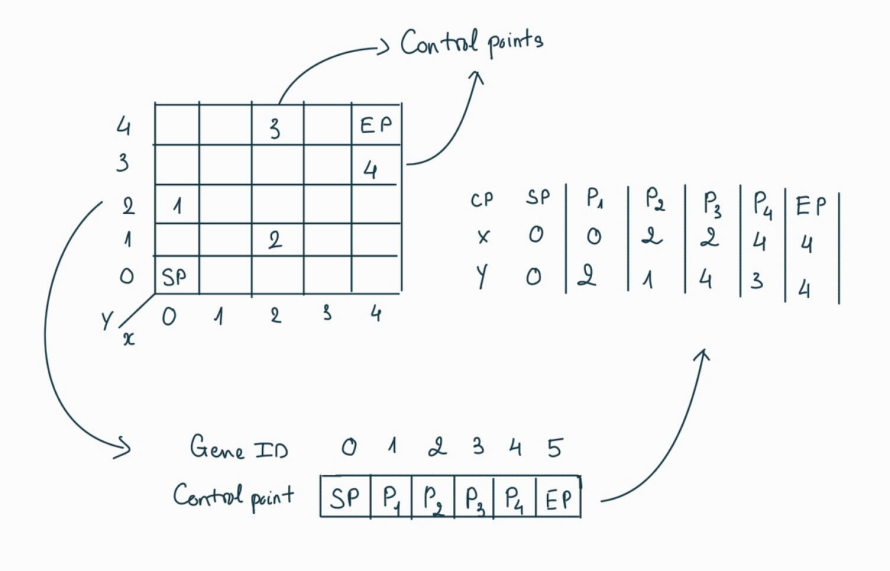


Figure 14: Chromosome Encoding Strategy

4.1.2 Danger Map Generation

For the static 2D map with size $H \times W$, with H and W are the height and width (in pixels) of the map respectively, we will compute the level of danger of each pixel on the map based on its distance from all surrounding obstacles

- (x, y): Coordinates of a pixel.
- $(O_{x,i}, O_{y,i})$: Coordinates of the i-th obstacle's center.
- r_i: Radius of the i-th obstacle (fully dangerous region).
- S: Vehicle size.
- $R_i = r_i + S$: Outer limit radius for the i-th obstacle.
- $d_i(x, y) = \sqrt{(x - O_{x,i})^2 + (y - O_{y,i})^2}$: Distance between the pixel (x, y) and the i-th obstacle.
- $D_i(x, y)$: Danger level contribution from the i-th obstacle.

The contribution of each obstacle to the danger level is calculated as follows:

$$D_i(x,y) = \begin{cases} 1, & \text{if } d_i(x,y) \le r_i, \\ 1 - \frac{\log_{10}(d_i(x,y) - r_i)}{\log_{10}(R_i - r_i)}, & \text{if } r_i < d_i(x,y) \le R_i, \\ 0, & \text{if } d_i(x,y) > R_i. \end{cases}$$

Cumulative Danger Level

The total danger level at a pixel (x, y) is the sum of danger level from all obstacles:

$$D(x,y) = \min\left(\sum_{i=1}^{N_{obs}} D_i(x,y), 1\right)$$

where N_{obs} is the total number of obstacles.

4.1.3 Initialize Population

At the start of the training procedure, an initial population needs to be generated. The number of individuals or population size C will be fixed throughout the training procedure. Each chromosome contains some initial amount of random genes.

4.1.4 Fitness Function

We defined the fitness values of the each chromosomes be the average danger level over the associated Bezier Curve by taking the line integral and then divided by the curve's length.

$$\mathbf{P} \Rightarrow B(t)$$

$$f_{fit}(B(t)) = \frac{\oint_L D(B(t))dl}{L} = \frac{\int_0^1 D(B(t))dt}{L}$$

where

- $B(t)$ is the Bezier curve parametric equation with respect to t converted from $\mathbf{P}$ which yields the coordinate of the local point with specific value of t
- L is the length of the Bezier curve

In computer programming, the finess values can be calculated by replacing the line integral with the sum of danger values of each discrete local point on the Bezier Curve divided by the number of points that we are calculating with. First, we create N_{bres} number of evenly spaced t values from ranging from 0 to 1. It is called "Bezier Resolution".

$$\mathbf{t} = \{0, 1 \times \frac{1}{N_{bres}}, 2 \times \frac{1}{N_{bres}}, 3 \times \frac{1}{N_{bres}}, ..., (N_{bres} - 1) \times \frac{1}{N_{bres}}, 1\}$$

After that, the average danger level of each chromosomes can be calculated as follows:

$$f_{fit} = \begin{cases} 1, \text{if } \exists O_i \text{ that } \|P_B(O_i) - O_i\| \leq r_i \\ \frac{1}{N_{bres}} \sum_{i=0}^{N_{bres}} D(\lfloor B(t_i) \rfloor) \end{cases}$$

The highest possible fitness value (most inefficient chromosome) returned is 1 when the chromosome crosses an obstacle. The $\lfloor B(t_i) \rfloor$ operation is used to round down the point's position, calculate the nearest pixel with integer value since the parametric

equation yields floating point values. The function $P_B()$ is used to find the current point's projection on the Bezier Curve, which we will define later in the Trajectory Tracking section.
After running through the finess function, the chromosomes will be separated into 2 groups: elites (fitness values $< T_S$) and non-elites (the rest) with T_S is the Separation Threshold.

4.1.5 Crossover and Mutation

Crossover

The crossover process will then be performed separately on 2 groups: elites and non-elites. In each group, we will select a proportion R_C (crossover ratio) of chromosomes to perform crossover. The process is defined as randomizing a crossover position on the interval $[0; min(L_{\mathbf{P_A}}, L_{\mathbf{P_B}})]$ with $L_{\mathbf{P_A}}$ and $L_{\mathbf{P_B}}$ are the lengths of 2 parent chromosomes respectively since the length of chromosomes throughout the training procedure may vary.
After that, the genes behind the crossover position of the 2 parental chromosomes will be exchanged.

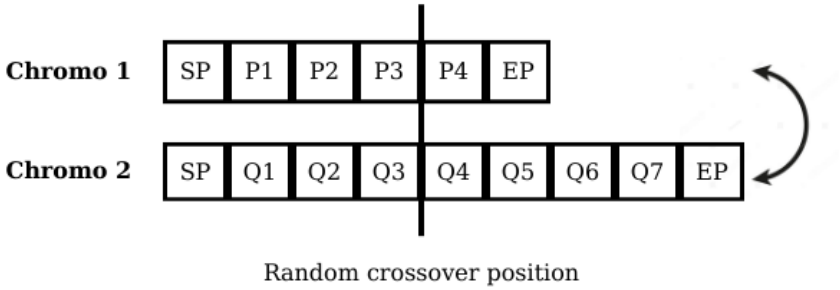


Figure 15: Crossover mechanism

Mutation

There are three types of mutation going to be used: add gene, remove gene and edit gene. The mutation process will be conducted only on a small proportion named Mutation Ratio R_M of the non-elite chromosomes. The mutation position will also be randomized.

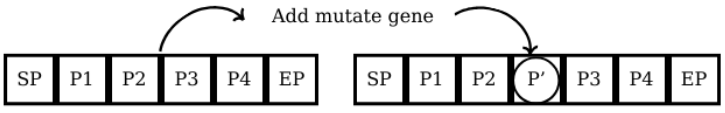


Figure 16: Add gene

The additional gene will be randomized in the scope of the map size

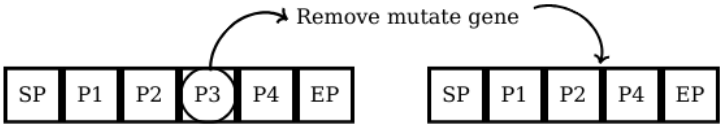


Figure 17: Remove gene

Randomly select a gene to be removed from the chromosome

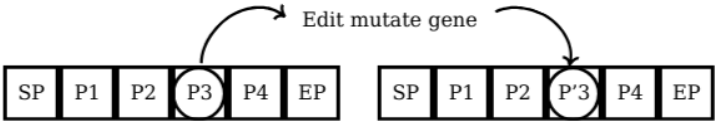


Figure 18: Edit gene

The chosen gene will be re-randomized in the scope of the map size.

4.1.6 Overal Algorithm Workflow

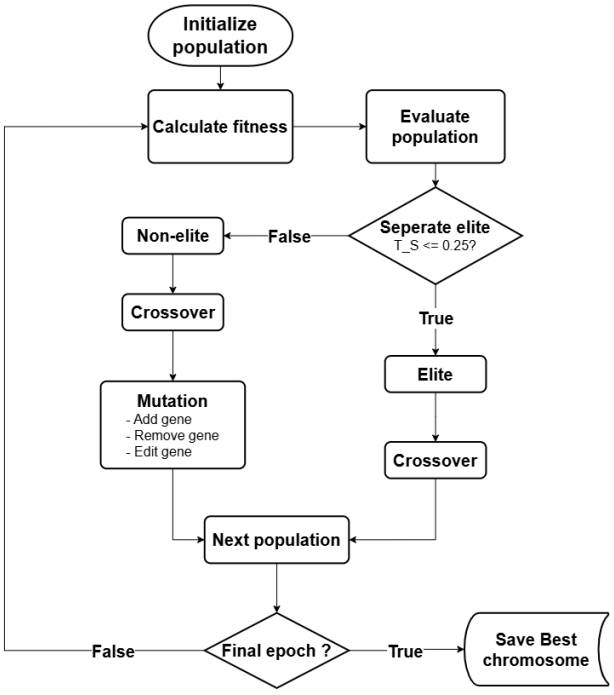


Figure 19: Algorithm Workflow

4.2 Trajectory Tracking Algorithm

The algorihm we introduce uses PID Controller with and the supporting Binary Search Algorithm to hhelp the car stays on track throughout the collison-free motion. There are several aspects need to be handle in order:

1. **Introduce the vehicle's model:** Propose the 2D motion of the car mathematically.
2. **Load the path**: The procedure starts by loading the best chromosome saved in the local memory that has been trained from the Genetic Model earlier. Then the chromosome will need to be converted into the Bezier Curve
3. **Control loop:**
    - Find the car's projection on the Bezier Curve
    - Calculate error
    - Calculate PID
    - Update vehicle's state

4.2.1 Vehicle model

The general motion equation of the car can be expressed with a set of equations:

$$X(t) = \int_0^t V_{car} \cdot \vec{u}(t)dt$$

$$\vec{u}(t) := R(\omega(t)dt) \cdot \vec{u}(t)$$

where:

- $\vec{u}(t)$ is a unit vector, representing the direction that the car is currently heading.
- $R(\alpha) = \begin{bmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{bmatrix}$ is the rotation matrix.

The position of the car with respect to time $X(t)$ is obtained by continously integrating velvelocity $V_{car} \cdot \vec{u}(t)$ with respect to time. To further simplify the problem, we define the speed of the car to be constant, hence the equation becomes:

$$X(t) = X_0 + V_{car} \cdot \vec{u}(t)dt$$

The heading vector $\vec{u}(t)$ at any time t is determined by continuously rotating the vector itself from the beginning of the simulation. The := operation means assigning the value back to the variable. In order to control the car, helping the vehicle follow the generated path, the introduced PID algorithm is utilized to calculate the angular velocity ω_{pid} each control loop.

However, in the simulated environment, all movements are represented through mathematical equations so the car would perform perfectly. In real-world scenarios, physical vibrations and inherent errors are inevitable. Therefore, incorporating these factors into the model is essential to validate the system's effectiveness.

$$\omega(t) = \omega_{pid} + \omega_{\Omega}$$

where ω_{Ω} is the system noise produced that we formulate as a representation of real world vibrations and errors that obeys the Gaussian Distribution.

$$\omega_{\Omega} = \eta \cdot x$$

with

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

where η is the maximum value of the the angular velocity noise can ever be achieved.

4.2.2 Initialization

- **Initial population:** Generating C chromosomes
- **Car position:** Coinciding the starting position.
- **Car heading:** Facing toward the ending position.
- **Load terrain:** Loading map with obstacles have been pre-created earlier.
- **Load best chromosome:** Executing the control points position of the chromosomes with the lowest fitness score throughout the epoches.
- **Convert chromosome to Bézier path:** Convert the saved chromosome into Bezier Path.

4.2.3 Control loop

Find projection

Determining the projection of a point on the Bezier Curve is the back bone principle of our PID control flow. Firstly, to minimize the computing time, we need to find a good starting point $B_{t_{mid}}$ on the curve before performing the Binary Search Algorithm by iterating over N_{bres} t values in $\mathbf{t}$.

$$t_{mid} = \arg\min_{t \in \mathbf{t}} \|B(t) - K\|$$

where K is the point that we are trying to find projection of.
Then we know for the fact that $B(t_{mid})$ is the closest point from K along the points in the Bezer resolution, the more optimal t value that yields a more optimal solution must

lies on the interval $\left[t_{mid} - \frac{1}{N_{bres}}; t_{mid} + \frac{1}{N_{bres}}\right]$.
Let us define $t_{left} = t_{mid} - \frac{1}{N_{bres}}$ and $t_{right} = t_{mid} + \frac{1}{N_{bres}}$ The algorithm works by repeatedly do the following operations in order:

1. Calculating the distance between K and the two middle points of the intervals $[t_{left}; t_{mid}]$ and $[t_{mid}; t_{right}]$

$$d_1 = \left\| B\left(\frac{t_{left}+t_{mid}}{2}\right) - K \right\| \text{ and } d_2 = \left\| B\left(\frac{t_{mid}+t_{right}}{2}\right) - K \right\|$$

2. Compare d_1 and d_2 to detemine which interval does the answer lies in
3. Ignore the interval that doesn't have the answer then update the left and right boundaries by assigning new values to t_{left} and t_{right}
   - $t_{right} = t_{mid}$ and keep t_{left} the same if $d_1 < d_2$
   - $t_{left} = t_{mid}$ and keep t_{right} the same if $d_1 \neq d_2$
4. Update $t_{mid} = \frac{t_{left}+t_{right}}{2}$
5. Check termination condition and repeat step 1 if not satisfied.
   The we define a Termination Threshold T_{BS} if $t_{right} - t_{left} \leq T_{BS}$ then the algorithm will yield the result $B(t_{mid})$

Calculate error

To steer the vehicle along the generated path, we first need to calculate it's position one step a head.

1. We define point Z be the predicted position of the car by calculating it's state using current calulcated data in the loop.
2. Then we find the projection P of Z on the Bezier Curve.
3. Then we calculate the error Y_{err}, which we will be using for the PID control.

$$Y_{err} = (\vec{u} \times \vec{ZP}) \cdot \|Z - P\|$$

Calculate PID

- **Propotional(P):** Adjusting the control output in proportion to the error magnitude.The larger the error, the stronger the corrective action.

$$P_{term} = P_I \cdot Y_{err}$$

- **Integral (I):** Addressing the accumulation of past errors. It is designed to eliminate steady-state errors that cannot be corrected by the proportional term alone.

$$I_{term} = K_I \cdot \int_0^t Y_{err}\, dt$$

- **Derivative(D):** Predicting the future behavior of the error by considering its rate of change. This term helps to dampen oscillations and improve stability.

$$D_{term} = K_D \cdot \frac{dY_{err}}{dt}$$

- **Control Variable ω_{pid}**: Minimizing deviations from a desired target or setpoint.

$$\omega_{\text{pid}} = K_P \cdot Y_{\text{err}} + K_I \cdot \int_0^t Y_{\text{err}}\, dt + K_D \cdot \frac{dY_{\text{err}}}{dt}$$

Update vehicle