KTH Royal Institute of Technology

Mechatronic Capstone course
MF2121

Can U-Turn

Spring term report

Authors:

Amy Annebäck	<amyan@kth.se>
Þórfríður Ina Arinbjarnardóttir	<tiar@kth.se>
Adrian Fasen	<fasen@kth.se>
Lewend Omar	<lewendo@kth.se>
Héctor Adriel Sanvicente Solís	<hass2@kth.se>
Panagiotis Charalampos Skoulaxinos	<pcsk@kth.se>
Yorkabel Yoans Tsegay	<yytsegay@kth.se>
Felix Wallberg	<fwallbe@kth.se>
Zhixuan Zhu	<zhixuanz@kth.se>

Supervisor:

Mikael Hellgren <hellgren@kth.se>

May 22, 2025

Contents

List of Figures

List of Tables

List of Acronyms

RC	Remote Controlled
SOTA	State Of The Art
CAN	Controller Area Network
IMU	Inertial Measurement Unit
GNSS	Global Navigation Satellite System
Radar	Radio Detection and Ranging
LiDAR	Light Detection and Ranging
IPM	Inverse Perspective mapping
LPS	Local Positioning System
GPS	Global Positioning System
ToF	Time of Flight
AoA	Angle of Arrival
RSSI	Received Signal Strengh Indicator
UWB	Ultra-Wideband
SLAM	Simultaneous Localization and Mapping
PF	Particle Filter
TDoA	Time Difference of Arrival
NLOS	Non-Line-of-Sight
KNN	K-Nearest Neighbour
CNN	Convolutional Neural Network
MCL	Monte Carlo Localization
RRT	Rapidly-exploring Random Tree
RRT*	Optimal Rapidly-exploring Random Tree
APF	Artificial Potential Field
SB-SQP	Sequential Bilinear Sequential Quadratic Programming
SQP	Sequential Quadratic Programming
S-NO	Sampling-based Nonlinear Optimization
DPP-CS	Dynamic Path Planning with Cubic Splines
HE*	Heuristic Enhanced A-Star

JQBC	Joint Quadratic Bézier Curves
FOBC	Fourth-Order Bézier Curves
PID	Proportional-Integral-Derivative
FLC	Fuzzy Logic Controller
SMC	Sliding Mode Control
MPC	Model Predictive Control
FMEA	Failure Mode and Effects Analysis

1 Introduction

In recent years, there has been a growing interest in autonomous systems, leading to rapid advancements in vehicle automation. As part of this trend, the project, titled "Can U-Turn", focuses on developing an autonomous Remote Controlled (RC) car capable of navigating a U-shaped track without any manual control. The goal is to create a fully self-driving system that serves not only as a technical challenge but also as a learning platform. The system will incorporate multiple aspects of autonomy, such as the car's localization, control, and management of the vehicle dynamics.

The report outlines the background of the project and the organizational structure of the team. It defines the project scope and the requirements that the final product has to meet. The results of the State Of The Art (SOTA) analysis are then summarized, followed by the concept design process. The report concludes with a reflection on the work carried out during the spring and then planned activities for the fall semester. Together, these sections provide a comprehensive overview of the development and design process involved in creating an autonomous RC car.

1.1 Background

This project is carried out within the scope of the Capstone Course for the Mechatronics master's programme at KTH Royal Institute of Technology in Stockholm. The course consists of 18 credits, 3 of which are allocated for the spring semester, during which the planning and SOTA analysis, together with parts of the system design, are performed. The remaining 15 credits are allocated for the fall semester, during which the project will be completed.

The stakeholder for this project is CanEduDev, a forward-thinking technology company focused on transforming how students learn the Controller Area Network (CAN) protocol. They have developed custom DevKits and rovers designed specifically for education, helping bridge the gap between theory and hands-on experience. The company created this project specifically for this Capstone Course, with the purpose of giving students practical knowledge in working with mechanical systems. CanEduDev has developed a RC car that allows for attaching different components implemented with the CAN communication protocol, which can be seen in Figure 1.1.

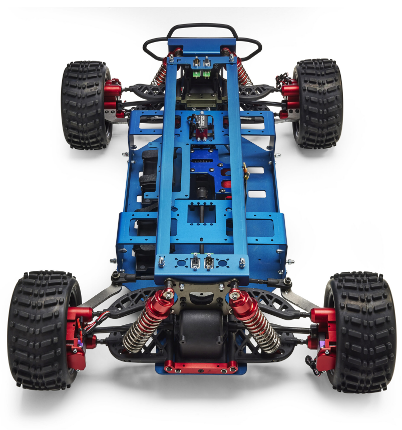
Figure 1.1: The RC car that CanEduDev has developed.

1.2 Organizational Structure

The project team comprises nine members currently enrolled in the mechatronics master's programme at KTH Royal Institute of Technology. Each group member comes from a different educational background, leading the team to have a broad knowledge base that allows for great exchanges of skills.

At the beginning of the term, a project leader was assigned, who organized the meetings and held each member accountable for their work. It was decided that the project team would meet weekly to review progress and to determine the next steps the team should take. In addition, weekly meetings with the stakeholders were held to keep them informed regarding the development of the project.

Initially, every group member conducted individual research regarding the project to determine what the SOTA could entail. Based on these findings, the team decided to structure the SOTA around four key aspects. Therefore, the project team decided to divide into four subgroups, where each group focused on one aspect. These subgroups carried out further reading and then presented their findings to the rest of the team, enabling the project team to determine together the final concept together.

1.3 Requirements

This section provides an overview of the functional requirements of the stakeholders. The requirements can be seen below:

- The autonomous RC-car must stay within the boundaries of the U-shaped track at all times.
- The autonomous RC-car must stop within the given goal area.
- The autonomous RC-car must finish the track within at most 60 seconds.

1.3.1 Test track

The autonomous RC car should be able to drive a U-shape track, as shown in Figure 1.2.

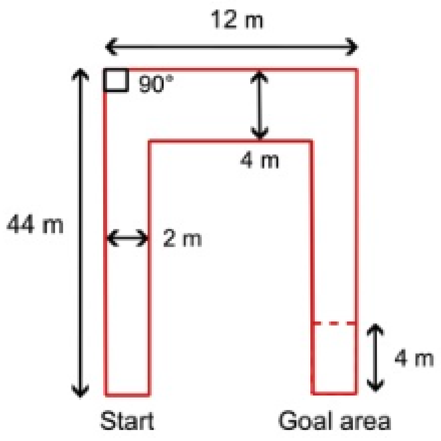

Figure 1.2: The U-shape track.

The track specifications are:

- Minimum track width: 2 meters
- Maximum track width: 4 meters
- Track length: 100 meters
- Short part of the U: 12 meters
- Goal area: Last 4 meters of the track
- Two 90-degree turns

The track will be located indoors, with cones marking its borders.

2 State Of The Art

In order to develop the autonomous RC car, several key components must be addressed. Four essential areas where identified for the SOTA research: perception, localization, path planning, and path following. These components form the foundation of any autonomous driving system, enabling the car to understand its surroundings, determine its position, plan a suitable path, and accurately follow it. Each area was investigated separately to evaluate current methods and determine the most appropriate solution for our project.

2.1 Perception

In this part of the SOTA, possible solutions for the perception of the autonomous RC car will be presented. Perception is how the car understands its surroundings. Sensors can be divided into two main categories: proprioceptive sensors, also called internal state sensors, which measure the internal values of a system such as angular rate, wheel load, etc., and exteroceptive sensors, also called external state sensors, which capture information about external values, such as distance measurements and light intensity. Some examples of proprioceptive sensors are Inertial Measurement Unit (IMU), encoders, inertial sensors such as gyroscopes, and positioning sensors such as Global Navigation Satellite System (GNSS) receivers. Some examples of exteroceptive sensors are cameras, Radio Detection and Ranging (Radar), Light Detection and Ranging (LiDAR), and ultrasonic sensors [1].

2.1.1 LiDAR

LiDAR is a sensing method that uses laser pulses to measure distance by calculating the time it takes for the laser to return after hitting an object. It provides highly accurate distance data and is commonly used for obstacle detection and mapping in autonomous systems.

LiDAR is particularly effective for precise distance measurement but is limited in recognizing or classifying objects. It is also relatively expensive compared to other sensing technologies [2].

2.1.2 Stereo and mono camera

Cameras produce clear images of an environment, including information about colour and texture. For an autonomous vehicle, cameras can help identify road signs, traffic lights, and other objects.

There are two main types of cameras: stereo and mono. A stereo camera uses two cameras to imitate the vision of animals, making it possible to sense depth using a stereo camera. The stereo camera compares disparities between points in the left and right images to create a disparity map, which is later converted to a depth map.

Mono cameras utilize a single camera, which is less expensive, but it limits the sensor's ability to measure depth. In order to measure depths using a mono camera, it is necessary to perform an Inverse Perspective mapping (IPM). This involves applying a Homography Matrix, H, which transforms image coordinates (u,v) into real-world coordinates (x',y'), as shown in Equation 2.1.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim H \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.1}$$

Some negatives of using cameras for perception are that the results can vary depending on the weather and that analysing image data can require significant computational power [1].

2.1.3 Local Positing System

Local Positioning System (LPS) are used to estimate the position of an object within a confined or indoor area where GNSS, such as Global Positioning System (GPS), are unreliable or unavailable. Unlike GNSS, LPS relies on a fixed infrastructure of reference points, known as anchors, installed in known positions throughout the environment. A receiver or tag mounted on the autonomous RC car communicates with these anchors to determine its position using techniques such as Time of Flight (ToF), Angle of Arrival (AoA), or Received Signal Strengh Indicator (RSSI) [3].

2.1.4 Radar

Radar is an old method for detecting surrounding objects. The method uses the Doppler property of electromagnetic waves to determine the objects' relative position and speed by emitting a wave and receiving the wave reflected off the object. There are three major categories of radars for autonomous vehicles: Short-Range Radar, Medium-Range Radar, and Long-Range Radar [1].

2.1.5 Ultrasonic sensors

Ultrasonic sensors measure distance by emitting high-frequency sound waves and calculating the time it takes for the echo to return after bouncing off nearby objects. They are commonly used in close-range applications due to their low cost and simplicity. In autonomous vehicles, ultrasonic sensors are particularly effective in detecting nearby obstacles during low-speed maneuvers such as parking. However, their performance can be affected by environmental conditions and has a limited range compared to other exteroceptive sensors [1].

2.2 Localization

Localization is fundamental to autonomous navigation. It allows a vehicle to determine its position and orientation within a known or partially known environment, which is essential for tasks such as path planning, obstacle avoidance, and control. Modern localization methods vary, from traditional sensor-based techniques to advanced probabilistic and machine learning models.

At their core, localization methods can be broadly categorized into relative and absolute techniques. Relative methods, such as odometry, estimate position by integrating motion over time using data from wheel encoders or IMU. Although low-cost and straightforward, these methods accumulate errors over time, making them unreliable for long-term navigation without some methodology for correction. Absolute localization techniques, like map-based localization or Ultra-Wideband (UWB) localization, rely on external references to maintain global accuracy. These techniques use known landmarks or features in conjunction with sensor inputs (e.g., LiDAR, camera, ultrasonic sensors) to localize the vehicle within a pre-mapped space [4].

This section presents and compares several typical localization approaches applicable to autonomous RC cars, including Simultaneous Localization and Mapping (SLAM), UWB localization, odometry, vision-based localization, Particle Filter (PF) based techniques, and map-based localization. Each method is discussed in terms of accuracy, complexity, sensor requirements, and suitability for small-scale autonomous platforms.

2.2.1 SLAM

SLAM is a fundamental technique used to explore an unknown environment. The exploration phase involves gathering data on an environment through sensors to discover its structure. A SLAM method performs two critical tasks concurrently: it supports the incremental building of a 3-D map representation of an environment while simultaneously using that same incremental map to localize the observer accurately. The purpose of using the map for localization is to minimize errors in the map-building process. This interconnected process of simultaneously mapping the environment and localizing within that developing map is the core mechanism of a SLAM system.

SLAM techniques can be implemented using a variety of sensors, but share the same principle. Some examples could be with LiDAR, mono camera, and a stereo camera. The accurate representation of the environment that results from a SLAM process can vary, potentially being a sparse set of landmarks or a detailed dense 3-D model, and it is considered highly useful for autonomous and human assistive applications [5].

2.2.2 UWB Localization

UWB localization is a short-range radio technology that uses a wide frequency spectrum [6]. UWB is a method which uses the LPS perception [3]. This method relies on time-based measurement techniques, such ToF, Time Difference of Arrival (TDoA), and two-way ranging. Applying trilateration to these measurements, the precise location of the tag can be determined [6].

The main advantages of UWB include high positioning accuracy, robustness, low power consumption, and minimal interference with other signals. However, it also presents challenges, particularly in Non-Line-of-Sight (NLOS) conditions and the need for careful anchor placements, as the coordinate system depends on the known location of these anchors [7].

2.2.3 Odometry-Based Localization

Odometry is one of the most fundamental localization techniques used in robotics. It involves estimating the vehicle's position and orientation (i.e., its "pose") by integrating motion data over time. To achieve this, wheel encoders are typically used and, in some cases, IMU.

Odometry estimates displacement by tracking the number of wheel rotations using encoders. By knowing the size of the wheels and the geometry of the vehicle, it becomes possible to compute the position and orientation in a 2D plane over time. This method is characterized by its simplicity, low cost, and computational efficiency, as it relies only on basic sensors and straightforward mathematical calculations. However, a key limitation of odometry is its susceptibility to cumulative errors. Minor inaccuracies in measurement, such as wheel slippage or uneven terrain, can accumulate over time, leading to increasingly inaccurate position estimates. Additionally, because odometry provides only relative positioning and lacks an external reference, it cannot independently correct these errors, making it unreliable for long-term localization without sensor fusion.

Due to these limitations, odometry is rarely used as a standalone localization method in complex or long-duration tasks. Instead, it is often fused with other sensors and methods to improve accuracy and robustness. In the context of autonomous RC cars, odometry remains a vital component, particularly in sensor fusion pipelines, offering real-time motion estimation that complements more sophisticated localization methods [8].

2.2.4 Vision-Based Localization

Vision-based localization, primarily employing cameras, constitutes a significant method in autonomous vehicle localization. An example of this would be the visual map creation and localization, which involves constructing a map from sequential images acquired along a trajectory, correlated with positioning data, often via GPS. To derive pose information, localization is subsequently performed by matching current camera images with this pre-existing visual map. Feature matching, often performed using machine learning methods such as K-Nearest Neighbour (KNN), is commonly used to compare image features. More advanced approaches apply deep learning methods like Convolutional Neural Network (CNN) to perform feature extraction and pose estimation for visual localization.

Camera-based vision methods are considered low-cost relative to sensors such as LiDAR. Reported localization accuracies vary considerably depending on the specific technique, map characteristics, and operational environment, with some studies achieving centimetre-level accuracy, while others report higher mean errors [4].

2.2.5 PF-Based Localization

Particle Filter (PF), encompassing approaches like the Monte Carlo Localization (MCL), represent a class of Bayes-filter-based methods commonly applied in autonomous vehicle localization systems. PFs are utilized primarily as a state estimation technique to determine the vehicle's pose and are frequently employed for fusing data from multiple sensors. In map-based localization contexts, PFs estimate the vehicle's position within a prior map by aligning observed sensor data with map features, adjusting particle weights based on the likelihood of observations given hypothesized poses. PFs are also applicable in mark-based localization, where they estimate pose by matching detected landmarks or road marks with their known locations within a sparse map. Integrating data from sensors such as LiDAR, IMU, GPS, wheel odometry, and cameras into the PF framework is a standard practice. Depending on the sensor inputs chosen, the PF location typically provides an excellent accuracy, ranging in the tens of centimetres [4].

2.2.6 Map-Based Localization

Map-based localization methods use a pre-built representation of the environment, such as a grid map, feature map, or 3D point cloud, to estimate a vehicle's position by comparing real-time sensor data against the map.

These methods typically rely on sensors like LiDAR, cameras, or ultrasonic sensors to perceive the surroundings.

Maps could take the form of occupancy grid, where the environment is divided into a grid of cells representing free, occupied, or unknown space, based on data from sensors such as LiDAR or ultrasonic sensors. Alternatively, maps can be topological, representing the environment as a graph of key locations (nodes) and the connections between them (edges), focusing more on the relationships between places than on exact distances. Topological maps are typically more compact and computationally efficient, and are well-suited for applications where high-level navigation decisions are needed rather than fine-grained control. Both maps provide a structured reference for localization algorithms to match real-time sensor observations with known environmental features, enabling the car to accurately estimate its position within the mapped space [4].

While map-based methods are robust and precise, they require accurate maps, sufficient sensor coverage, and relatively high computational resources. As such, they are often combined with odometry, SLAM, or PF systems in autonomous RC car platforms to achieve reliable navigation.

2.3 Path planning

This section of the SOTA will explain path planning as a solution to how an autonomous vehicle can find its way towards a goal. Path planning is important because if the vehicle deviates too much from the expected path, or if an obstacle suddenly appears, the vehicle might have to recalculate how to get back to the intended path and the goal [9]. Path planning algorithms in the autonomous driving system, a comprehensive review by Mohamed Reda et al suggests five main methods of obtaining a global and/or local path: graph-based methods, sampling-based methods, gradient-based methods, optimization-based methods, and interpolation curve methods. Each is described in the following subsections.

2.3.1 Graph-based methods

A commonly used traditional method of path planning is graph-based path planning. A graph is a mathematical structure that contains nodes - points or locations in space, such as pixels in an image - as well as arcs, which count as the space connecting these nodes, representing possible movements, along with possible associated costs like distance [10]. Obtaining graphs for path planning can also be divided into two methods; global (offline) planning, where the graph or map is pre-generated and supplied at the start of testing - or local (online) planning, where the graph is continuously updated with the help of sensors and localization/mapping techniques.

A study by Menaka Naazare et al. outlines the advantages and disadvantages of online graph-based path planning methods, saying that while localization is still accurate, their UAV was able to follow intended trajectories using the A* algorithm successfully. They also cited a study by Parikshit Maini and P.B. Sujit that successfully used the Dijkstra algorithm for simulations [11]. As such, one can argue that if graph-based path planning is sufficient for aerial vehicles, it should also suffice for most other vehicles, such as autonomous cars. A notable disadvantage also brought up in their study was a heavy reliance on GPS data, or a dependency on accurate and frequent updates in location. A strong localization method would be recommended for online graph-based planning, as this method needs to recreate the graph from scratch many times over a single path to account for errors and sensors not being able to supply a complete graph for the intended movement.

2.3.2 Sampling-based methods

Sampling-based planning is easily confused to be the same as graph-based planning using online planning. This is because sampling-based methods continuously and randomly generate nodes from environment samples. The randomly selected path can be checked for obstacles and expanded upon until it becomes feasible to reach the goal. Therefore, unlike graph-based methods, sampling-based methods don not need to generate a full map of the surroundings, but only generate enough nodes to find a path to the goal [12].

A study on motion planning by Christos Katrakazas et al. describes the top advantage of Rapidly-exploring Random Tree (RRT), which is a type of sampling-based method, to be how fast it can find a viable path because that it does not need to take any further action once it finds a single path that successfully leads to the goal. It is also outlined, though, that this path can be suboptimal in terms of time and distance, as well as being jerky and rough due to the randomness involved in generation [13]. A way to solve the suboptimal pathing of RRT is with the improved method, Optimal Rapidly-exploring Random Tree (RRT*). RRT* does not stop after finding a viable path, but keeps improving on the current tree. RRT* will converge on the optimal path as the number of samples generated goes to infinity [12].

2.3.3 Gradient-based methods

Gradient-based methods differ from the previous two. An example of a gradient-based method is Artificial Potential Field (APF). To plan the path, APF sees the goal point as an attractive force and all obstacles as repelling forces. It then combines the effects of all forces into a field and follows where the gradient of the field is pointing. APF does not necessarily need a map or graph to find the goal, but it does need to know the coordinates of the goal to get going. Benefits of this method are that it can find paths that avoid all obstacles within a short computational time window, with the drawbacks that the chosen path around an obstacle may be suboptimal, and that the algorithm stops completely if the gradient ever becomes zero, as there would then be no pull in any direction [9].

As discussed in a study by Yu Li et al., the Local Minima Trap is a common challenge with APF. Local Minima Trap occurs when all attractive and repulsive forces sum up to zero, meaning the gradient is zero, before reaching the goal destination, prematurely stopping all movement. The authors argue that there are several ways to solve this issue, such as with the use of perturbation - introducing random walking patterns to get out of the local minima [14].

While all these kinds of methods have many different algorithms available for use, it is notable that a vast collection of these algorithms are built on the same base algorithms, such as A*, RRT and APF [9], because these algorithms are a good starting point that may need a minor alteration to function in the intended way for a given project.

2.3.4 Optimization-based methods

Optimization-based methods (mathematical programming) are used to find numerical solutions to optimization problems. These methods start by building an objective function based on the path planning requirements. The goal is to find the maximum or minimum value of this function. When minimizing the objective function, the problem is called convex programming. When maximizing it, the problem is called concave programming [15].

If the objective function includes a quadratic term, the method is called quadratic programming [16]. Path planning can be modelled in autonomous driving as a quadratic programming problem where the goal is to minimize the path length and find the best route.

When the objective function includes random variables or the input during the search process is random, the problem is called a stochastic optimization problem. This kind of problem can also be solved using dynamic programming [17].

Optimization-based methods can also be combined with other techniques. For example, the Sequential Bilinear Sequential Quadratic Programming (SB-SQP) algorithm is a hybrid method that combines sampling-based algorithms with Sequential Quadratic Programming (SQP) [18]. Similarly, the Sampling-based Nonlinear Optimization (S-NO) algorithm uses sampling techniques and quadratic programming to solve convex optimization problems [19].

2.3.5 Interpolation curve methods for path planning in automated driving systems

The interpolation curve method is a way to create a new path from an existing one to avoid collisions. At first, there is an initial path, but this path may collide with an obstacle. The interpolation method changes the original path and creates a new sub-path that is safe and smooth. This kind of path planning uses special curves. The most common curves are Bezier curves [20], Clothoid curves [21], Splines curves [22], and Polynomial curves[23].

Based on different curves and techniques, researchers have developed many algorithms. Some examples include the Dynamic Path Planning with Cubic Splines (DPP-CS) algorithm curves[24] (which uses cubic spline interpolation), the Heuristic Enhanced A-Star (HE*) algorithm, the Joint Quadratic Bézier Curves (JQBC) algorithm curves[25], and the Fourth-Order Bézier Curves (FOBC) algorithm curves[26].

As a stand-alone method, the interpolation method is slow and may not work well in complex situations. However, when combined with other methods, such as graph-based or sampling-based algorithms, it can work faster and create better, more useful paths.

2.4 Path Following

This section discusses the path tracking module, as referred in literature, of the autonomous navigation system [27]. Through this report it will be named as fath following for clarity. This module takes the path generation module output, this being a series of waypoints or a continuous mathematical function in the vehicle's coordinate system [28], and generates the steering, accelerating, and braking references for the low-level controllers that drive the actuators [29]. In resume, the path tracking module takes care of two tasks [30]:

1. Lateral control: Drives the steering to stay within the desired racing line given as input with minimal error.
2. Longitudinal control: It is responsible for following the appropriate velocity profile along the path.

Historically, control methods for trajectory tracking in autonomous vehicles have been categorized into geometric algorithms, kinematic model controllers, and dynamic model controllers [31]. However, three categories are identified for this SOTA: Kinematic controllers, classical controllers, and robust controllers.

Kinematic controllers based purely on kinematic models and geometric relationships, ignoring dynamic effects, which makes them easy to implement. Classical controllers use traditional, typically linear, control methods that can handle system dynamics if properly modelled, offering good performance with relatively simple implementation, however, they are less effective in the presence of non-linearities, uncertainties, or disturbances. Robust controllers are designed to handle such challenges by accounting for uncertainties and external disturbances, delivering high performance at the cost of increased complexity and computational load.

2.4.1 Pure pursuit - Kinematic controller

Pure Pursuit is a geometric trajectory tracking algorithm based on Ackerman's steering geometry for a bicycle model [32]. It uses the geometric information of the kinematic model to calculate a steering angle that follows an arc to a point, called the look-ahead point, located on the reference trajectory. The control law is calculated following Equation 2.2.

$$\delta = \arctan\left(\frac{2L\sin(\alpha)}{l_d}\right) \tag{2.2}$$

The Pure Pursuit controller uses the look-ahead point located at a fixed distance l_d ahead of the vehicle's current position to compute the required steering angle. This angle remains constant until a new point enters the look-ahead range, prompting a recalculation [33].

The parameters L and α represent the vehicles' wheelbase and the angle between the vehicle's heading and the look-ahead point, respectively. Since the steering law is independent of velocity, variations in speed can lead to inconsistent behaviour, producing overly aggressive or sluggish steering responses. To address this, a dynamic look-ahead distance proportional to the velocity ($k_v v_x$) has been proposed to improve stability across different speeds [34].

Pure Pursuit is simple to implement and computationally efficient thanks to its purely kinematic formulation. However, ignoring dynamic factors like friction or lateral forces can reduce control accuracy or introduce instability. The controller's performance heavily relies on the quality of the reference trajectory, velocity control, and appropriate tuning. When poorly tuned, it may exhibit behaviours such as corner-cutting, especially problematic in environments with strict boundaries [35].

2.4.2 Stanley - Kinematic controller

The Stanley method is a geometric path-tracking algorithm based on the Ackerman kinematic bicycle model, and Stanford University employed it to win the 2005 DARPA Grand Challenge, which is an autonomous vehicle rally that takes place in the United States. Unlike look-ahead-based controllers, Stanley uses a proportion k_s to the current lateral error, e, to the reference path, the heading error, $\theta = \psi_c - \psi_{\text{traj}}$, where $(\psi_c, \psi_{\text{traj}})$ are orientation of the car and the corresponding point in the path for that time, as well as the velocity v_x to calculate the steering angle using Equation 2.3 [31]:

$$\delta(t) = \theta + \arctan\left(\frac{k_s e}{v_x}\right); \delta \in [-\delta_{\max}, \delta_{max}] \tag{2.3}$$

As a geometric method, Stanley is computationally efficient and relatively easy to implement, being widely supported by off-the-shelf tools such as MATLAB and Simulink [36]. However, the basic version of the algorithm may struggle with discontinuous paths or significant tracking errors [35]. The complete version of the Stanley controller, as used in the DARPA Challenge, includes additional terms to account for lateral dynamics and improve stability, as shown by Hoffman et al [37].

This controller is inherently reactive, meaning it corrects the trajectory only after an error is detected, as it lacks a look-ahead or feed-forward component, which often lead to overshooting the track [31].

Recent improvements have included predictive elements to compensate for the lack of look-ahead [38], and hybrid approaches integrating Proportional-Integral-Derivative (PID) control to enhance robustness while retaining Stanley's simplicity [28].

2.4.3 PID controller - Classical controller

A classical Proportional-Integral-Derivative (PID) controller is widely used to solve the path tracking problem in autonomous vehicles [31]. The control input is typically based on the cross-track error, defined as $e = y_d - y$ where y_d is the desired lateral position at time t, and y is the vehicle's current lateral position. The PID control law is expressed as:

$$\delta = K_p e(t) + K_i \int e(t)dt + K_d \frac{d}{dt} e(t) \tag{2.4}$$

When applied specifically to path tracking, the behaviour of each term changes slightly due to the dynamic nature of the reference trajectory when compared to a normal PID operation: The derivative term K_d becomes particularly valuable, as it enhances the controller's responsiveness to rapidly changing paths; the proportional gain K_p is still essential for maintaining closed-loop stability, though it contributes less to precision in tracking. In contrast, the integral term K_i often becomes ineffective or even detrimental, since the continuously changing target position prevents the error from stabilizing over time [30].

PID controllers are widely used due to their simplicity and the fact that they do not require a detailed system model, making them easy to implement and tune. However, their performance can degrade under changing system dynamics, as they are sensitive to environmental variations that differ from the tuning conditions [32]. Depending on gain settings, they may also exhibit issues such as chattering, overshooting, or dead-band behaviour. To overcome these limitations, adaptive approaches using fuzzy logic or neural networks have been developed, enabling the controller to adjust its parameters dynamically during runtime [39, 40].

2.4.4 Fuzzy controller - Classical controller

Fuzzy Logic Controllers (FLCs) are rule-based systems that use fuzzy sets to model uncertainty and approximate reasoning through a collection of intuitive "if-then" rules instead of relying on a precise mathematical model [41]. Therefore, FLCs are particularly useful for complex, non-linear, or poorly defined systems. Their design allows for high flexibility and robustness, especially in environments where traditional control techniques struggle, such as driving backwards [42].

Fuzzy controllers are relatively simple to integrate and require modest computational resources, making them viable for real-time applications and Off-the-shelf tools and libraries—such as MATLAB's Fuzzy Logic Toolbox—facilitate development and deployment [43]. However, fuzzy logic controllers can suffer scalability issues when dealing with high-dimensional input spaces, as the rule base may grow exponentially with the number of variables [41].

Additionally, the tuning of membership functions and rule sets is often heuristic and may require expert insight or optimization methods [31]. Despite these challenges, FLCs remain a popular choice in commercial and academic applications due to their balance of simplicity, adaptability, and reliability [43, 42].

2.4.5 Sliding Mode Control - Robust controller

Sliding Mode Control (SMC) is a non-linear control technique that guides the system state toward a predefined path in the state space, known as the sliding surface, so that the control becomes simpler and more predictable, ensuring stable behaviour [44]. SMC is considered a robust control as it is inherently capable of handling disturbances, uncertainties, and changing system parameters without significantly affecting performance, which makes it particularly useful for systems with modelling inaccuracies and where fast and reliable response is essential, like path following [32].

However, SMC is not without challenges. A common issue is chattering, which is rapid-small oscillations in the control signal that can occur near the sliding surface. This effect worsens when the controller gains are increased, and as a result, tuning an SMC controller can be difficult [45]. Another concern is that if the reference path changes suddenly, the controller may give large inputs, producing potentially dangerous levels of lateral acceleration [31].

To overcome these limitations, improvements have been proposed, such as using adaptive sliding mode control with a fuzzy logic observer, where fuzzy rules adjust the controller response based on how far the system is from the sliding surface, reducing chattering while maintaining robustness [46]. Another strategy involves integrating neural networks, which can learn and compensate for disturbances, allowing the controller to use less aggressive settings [47].

2.4.6 Model Predictive Control - Optimization-based controller

At its core, Model Predictive Control (MPC) uses a mathematical model of a system to predict the behaviour of the system over a defined time horizon and formulates a control law by solving an optimization problem to minimize a cost function considering constraints on control inputs and states [48]. It solves the optimization problem at each time step, applying only only the input for the first time step of the prediction horizon. The process is then repeated at the next time step with the updated state information [34].

The main issue with MPC is that it is very computationally complex, as it takes into account the model, cost function, predicting future states, and constraints. Therefore, this requires more expensive hardware, or alternatively longer time-steps, which in turn reduces performance. The complexity of the model also plays a big role in computational load. If the model is highly non-linear, it adds another layer of complexity when solving the optimization problem.

3 Concept Design

Key components covered in the SOTA analysis, chapter 2, were evaluated to define a suitable system architecture for the autonomous RC car. The SOTA analysis focused primarily on software-based functionality and the selection of sensors required to support it. Therefore, hardware elements like motors, the mechanical structure and, vehicle dynamics were not central to this this concept design.

Several methods were compared uding dedicated evaluation matrices for For each component within the SOTA analysis. The criteria used in each matrix were tailored to the nature of the component, ensuring a meaningful comparison. A more detailed explanation of the evaluation metrics can be seen in Appendix A, and an explanation of the criteria can be seen in Table A.1.

The following sections provide a breakdown of the evaluation for each component, the criteria used, and the rationale behind the selected methods. They also go over mechanical considerations that will be need to be taken into account later in the development phase.

3.1 Perception

The perception technologies evaluated were LiDAR, stereo camera, mono camera, LPS, radar, and ultrasonic sensors. The criteria included computational simplicity, adaptability, monetary cost, consistency, accuracy, implementation simplicity, and resolution. Each method was scored out of a maximum of 35 points; the scoring matrix can be seen in Table A.2.

LiDAR scored the highest (29) in the matrix due to its adaptability, consistency, accuracy, and high resolution. LPS (27) followed closely, as it performed well across the criteria but lacked computational simplicity. Mono and stereo cameras scored reasonably (24 and 22, respectively), offering good adaptability and high resolution. However, they suffer from low implementation simplicity because they require complex image processing algorithms, contributing to increased computational demands. Radar and ultrasonic sensors ranked lowest (19 and 17, respectively) primarily due to limitations in resolution, consistency, and accuracy.

LiDAR stands out as the most robust perception option for detailed and accurate environmental understanding, provided that its computational and integration requirements can be met.

3.2 Localization

The localization methods evaluated were SLAM, UWB, odometry, vision-based, PF, and map-based. Criteria included adaptability, consistency, accuracy, computational simplicity, and implementation simplicity. Each method was scored out of a maximum of 30 points; the scoring matrix can be seen in Table A.3.

All methods obtained relatively similar scores, ranging from 17 to 20 points. Map-based localization scored the highest (20), performing strongly across most criteria, particularly on consistency, accuracy, and implementation. PF followed closely (18), excelling in adaptability and accuracy, but scoring lower on computational and implementation simplicity. The remaining methods, SLAM, UWB, odometry, and vision-based localization, scored 17 points. Despite different strengths, such as high accuracy in vision-based methods or strong computational and implementation simplicity in odometry, each had trade-offs that balanced their overall scores.

The PF, map localization, and odometry were chosen for the final concept. Although odometry tends to accumulate error over time, it offers valuable short-term motion data that complements the other methods well. PF and map-based localization can complement this by offering high accuracy and consistency, helping to correct and stabilize the accumulated error. Combining these methods creates a strong candidate for a robust localization method.

3.3 Path Planning

The path planning methods evaluated were graph-based, sampling-based, gradient-based, optimization-based, and interpolation curve. Criteria included adaptability, consistency, path optimization, computational simplicity, and implementation simplicity. Each method was scored out of a maximum of 25 points; the scoring matrix can be seen in Table A.4.

Graph-based planning achieved the highest overall score (19),clearly balancing consistency and implementation simplicity. Interpolation curve (18) and gradient-based planning (17) methods followed closely behind. The interpolation curve performed relatively even across most criteria, benefiting from computational simplicity and gradient-based planning showed strong consistency and adaptability. Sampling-based and optimization-based planning received the fewest points (both receiving 15). However, optimization-based planning performed best in path optimization but is more computationally and implementation complex.

However, when reviewing common methods of path planning, it was decided that there may not be a need to continuously update the planned path of the car. The track is static, and there would be no unexpected obstacles. Therefore, only a global path would be sufficient, assuming any deviations from the intended path will be minor. To maximize robustness, a path planning algorithm will be applied initially to compute the optimal path to follow, and it can be reapplied during operation if the vehicle drifts too far from the planned trajectory.

Therefore, a combination of graph-based and optimization-based planning will be used. Optimization-based planning will be used to find the initial optimal path for the autonomous RC car to follow. In case of large deviations, graph-based methods, such as A* and Dijkstra, offer a simple and consistent approach to generating a new feasible path. This choice aligns with the evaluation results, where graph-based planning scored highest overall, and optimization scored highest in the path optimization criteria.

3.4 Path Following

The five path-following control methods presented in Section 2.4 were evaluated: Pure Pursuit, Stanley, PID, Fuzzy, SMC, and MPC. Criteria included accuracy, robustness, reliability, high-speed capability (>2 m/s), computational simplicity, and implementation simplicity. Each method was scored out of a maximum of 30 points; and the evaluation matrix can be seen in Table A.5.

Among the evaluated options, Stanley (27), Pure Pursuit (25), and PID (22) stood out due to their computational and implementation simplicity. Stanley achieved the highest score, receiving high marks across all criteria. Pure Pursuit and PID followed Stanley closely, where they were valued for their simplicity and reliability. MPC (23) showed high performance in the first four criteria mentioned, but was limited by computational and implementation complexity. SMC (20) and Fuzzy (19) scored lower because they received more moderate scores for each criterion.

Based on the evaluation matrix, a hybrid control strategy was selected. The Stanley method will be used for lateral control, and PID control will be used for longitudinal dynamics. PID was chosen over Pure Pursuit because the team had more experience with this method.

3.5 Mechanical Considerations

Although the primary focus of the concept design has been on the software and the autonomous RC car's ability to drive the track, the mechanical aspects of the implementation have been considered. It is known that CanEduDev will provide the car chassis. However, it might require some modifications, as the stakeholder only raised concerns later in the design process regarding its suitability to be able to drive the track. Therefore, the mechanical decisions that are still under consideration include:

- Whether to use front-wheel drive, rear-wheel drive, or four-wheel drive.
- The use of independent motors for each wheel versus a single-motor drivetrain.
- Redesigning steering mechanisms to be able to go through the 90-degree turns. Furthermore, potentially adding torque vectoring and/or rear-wheel steering.

These choices will directly affect the ability of the autonomous RC car's ability to drive the given path and will be refined further during the implementation phase.

3.6 Final concept summary

Based on the comparative analysis across all the core components of the system, the following methods were selected for the autonomous RC-car concept:

- **Perception:** LiDAR will be used due to its high resolution and accuracy.
- **Localization:** Particle filter, map-based localization, and odometry will be used together to balance robustness and simplicity
- **Path planning:** Graph-based and optimization-based methods will be used to find the optimal path.
- **Path following:** PID control will be used for the longitude control, and Stanley will be used for lateral control.

The final concept prioritizes accuracy and robustness while maintaining computational simplicity. It draws from well-established methods supported by the evaluation results. Combining complementary techniques within each function, allows the system to perform reliably under the expected conditions. Therefore, the final concept establishes a solid foundation for further development and testing in the upcoming phases.

4 Discussion

This chapter provides a broader reflection on key components from the work conducted. Firstly, the final concept design is revisited, and the trade-offs and feasibility of the final concept are discussed. It is then followed by a risk analysis outlining the potential challenges anticipated during the development and testing phase. Finally, the chapter concludes with a look ahead to the fall semester, outlining future work and the organizational structure the team will follow.

4.1 Design concept

The final concept results from carefully balancing performance, feasibility, and available resources. However, the concept is built around a set of assumptions regarding the autonomous RC car's mechanical and software capabilities. In particular, it assumes that the provided chassis from CanEduDev will be sufficient for the car to drive the test track, an assumption that was challenged late in the process, when the stakeholders raised concerns about the chassis steering configurations and the car's ability to handle sharp turns. As a result, these mechanical constraints were not a primary focus during the design process, but may significantly affect the car's driving ability.

Throughout the concept design, one of the main trade-offs in the concept evaluation was between performance and computational complexity. Due to the limited financial and time resources, it was considered more practical to opt for less computationally heavy algorithms, even if this meant giving up some performance. The final concept design might have looked different if the concept were solely based on maximizing performance.

The current control strategy, a hybrid control strategy using the Stanley method for lateral control and PID control for longitudinal dynamics, was chosen to meet the project requirements while remaining computationally simple. However, it relies heavily on the assumption that the car's mechanical constraints, such as the steering angle limits, are not a significant issue. If this assumption proves invalid, the current control strategy may need reconsidering. In such cases, more advanced methods that account for nonlinearity and uncertainty, such as Fuzzy logic, SMC, or MPC, could offer better performance and robustness.

While the SOTA primary focus was on the software components, the mechanical design of the car will play an equally crucial role in its performance. Whether these, somewhat simpler, methods will be sufficient is something that will be revealed in simulations and real-world testing. Their success will largely depend on how the subsystems integrate and perform. The flexibility of the system architecture will play a big role in allowing for iterative refinement as development progresses.

4.2 Risk Analysis

Risks are inevitable when working on complex projects such as developing an autonomous RC car. These risks can range from technical failures to communication issues. A risk analysis was conducted using the Failure Mode and Effects Analysis (FMEA) to address these potential risks proactively. This structured approach allowed for systematically identifying and evaluating probable causes of risks, assessing their consequences, and determining both preventative and reactive measures. The risks were prioritized using a likelihood-severity matrix. This matrix evaluates each risk on the probability of its occurrence and the severity of its impact. The results are categorized into different levels, ranging from Safe to Unacceptable. The whole risk analysis matrix as well as the likelihood-severity matrix can be seen in Appendix B, which outlines each risk and its corresponding preventative and reactive measures strategies.

4.3 Future works

The fall plan was initially organized around four technical subgroups: perception, localization, path planning, and control, mainly software-focused tasks. However, the team and task planning were restructured once it became evident that vehicle dynamics would play a significant role in achieving a functional system. The detailed time plan for the fall can be seen in Appendix C

The various tasks in the time plan are divided according to the main project phases, including development, subsystem testing, integration, and evaluation. This schedule includes important administrative activities such as report writing, presentations, and demonstrations.

4.3.1 Organization

In the fall, the project team will divide into the following three focused subgroups:

- **Vehicle Dynamics:** Felix Wallberg, Panos Skoulaxinos, Þórfríður Arinbjarnardóttir
- **Localization:** Yorkabel Tsegay, Amy Annebäck, Adrian Fasen
- **Control:** Zhixuan Zhu, Héctor Sanvicente, Lewend Omar,

After dividing into these subgroups, the team plans to adopt an agile workflow incorporating sprints. This approach supports continuous progress, frequent integration, and flexibility in responding to feedback throughout the development phase.

Bibliography

[1] De Jong Yeong et al. "Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review". In: *SENSORS* 21.6 (Mar. 2021). Num Pages: 37 Place: Basel Publisher: MDPI Web of Science ID: WOS:000652714300001, p. 2140. ISSN: 1424-8220. DOI: 10.3390/s21062140. URL: https://www.mdpi.com/1424-8220/21/6/2140 (visited on 05/19/2025).

[2] You Li and Javier Ibanez-Guzman. "Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems". In: *IEEE Signal Processing Magazine* 37.4 (July 2020), pp. 50–61. ISSN: 1558-0792. DOI: 10.1109/MSP.2020.2973615. URL: https://ieeexplore.ieee.org/document/9127855 (visited on 04/14/2025).

[3] Faheem Zafari, Athanasios Gkelias, and Kin Leung. *A Survey of Indoor Localization Systems and Technologies*. Jan. 16, 2019. DOI: 10.48550/arXiv.1709.01015. arXiv: 1709.01015[cs]. URL: http://arxiv.org/abs/1709.01015 (visited on 05/21/2025).

[4] Debasis Kumar and Naveed Muhammad. "A Survey on Localization for Autonomous Vehicles". In: *IEEE Access* 11 (2023), pp. 115865–115883. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3326069. URL: https://ieeexplore.ieee.org/document/10287946/ (visited on 05/19/2025).

[5] "Exploration: Simultaneous Localization and Mapping (SLAM)". In: Samunda Perera, Dr.Nick Barnes, and Dr.Alexander Zelinsky. *Computer Vision*. Boston, MA: Springer US, 2014, pp. 268–275. ISBN: 978-0-387-30771-8 978-0-387-31439-6. DOI: 10.1007/978-0-387-31439-6_280. URL: http://link.springer.com/10.1007/978-0-387-31439-6_280 (visited on 05/17/2025).

[6] Lucija Bukvić et al. "Ultra-wideband Localization with Time-based Measurement Techniques". In: *ResearchGate*. Sept. 13, 2023. DOI: 10.1109/ELMAR59410.2023.10253907. URL: https://www.researchgate.net/publication/374107169_Ultra-wideband_Localization_with_Time-based_Measurement_Techniques (visited on 05/18/2025).

[7] Boyang Sun et al. "A Reliable and Efficient UWB-Based Indoor Localization Scheme". In: *2023 IEEE 15th International Conference on Advanced Infocomm Technology (ICAIT)*. 2023 IEEE 15th International Conference on Advanced Infocomm Technology (ICAIT). ISSN: 2770-1603. Oct. 2023, pp. 186–191. DOI: 10.1109/ICAIT59485.2023.10367261. URL: https://ieeexplore.ieee.org/document/10367261 (visited on 05/18/2025).

[8] Yuan Zhuang et al. "Multi-sensor integrated navigation/positioning systems using data fusion: From analytics-based to learning-based approaches". In: *Information Fusion* 95 (July 2023), pp. 62–90. ISSN: 15662535. DOI: 10.1016/j.inffus.2023.01.025. URL: https://linkinghub.elsevier.com/retrieve/pii/S1566253523000398 (visited on 05/19/2025).

[9] Mohamed Reda et al. "Path planning algorithms in the autonomous driving system: A comprehensive review". In: *Robotics and Autonomous Systems* 174 (Apr. 1, 2024), p. 104630. ISSN: 0921-8890. DOI: 10.1016/j.robot.2024.104630. URL: https://www.sciencedirect.com/science/article/pii/S0921889024000137 (visited on 05/18/2025).

[10] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (July 1968), pp. 100–107. ISSN: 2168-2887. DOI: 10.1109/TSSC.1968.300136. URL: https://ieeexplore.ieee.org/stampPDF/getPDF.jsp (visited on 05/18/2025).

[11] Menaka Naazare et al. “Application of Graph-based Path Planning for UAVs to Avoid Restricted Areas”. In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). ISSN: 2475-8426. Sept. 2019, pp. 139–144. DOI: 10.1109/SSRR.2019.8848968. URL: https://ieeexplore.ieee.org/document/8848968 (visited on 05/18/2025).

[12] Andreas Orthey, Constantinos Chamzas, and Lydia E. Kavraki. *Sampling-Based Motion Planning: A Comparative Review*. Sept. 22, 2023. DOI: 10.48550/arXiv.2309.13119. arXiv: 2309.13119[cs]. URL: http://arxiv.org/abs/2309.13119 (visited on 05/18/2025).

[13] Christos Katrakazas et al. “Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions”. In: *Transportation Research Part C: Emerging Technologies* 60 (Nov. 1, 2015), pp. 416–442. ISSN: 0968-090X. DOI: 10.1016/j.trc.2015.09.011. URL: https://www.sciencedirect.com/science/article/pii/S0968090X15003447 (visited on 05/18/2025).

[14] Yu Li et al. “Path Planning of Unmanned Ships Using Modified APF Algorithm”. In: *2023 IEEE 11th International Conference on Computer Science and Network Technology (ICCSNT)*. 2023 IEEE 11th International Conference on Computer Science and Network Technology (ICCSNT). ISSN: 2690-5892. Oct. 2023, pp. 112–119. DOI: 10.1109/ICCSNT58790.2023.10334590. URL: https://ieeexplore.ieee.org/document/10334590 (visited on 05/18/2025).

[15] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer, 1999.

[16] Noor Alhuda F. Abbas. “Mobile Robot Path Planning Optimization Based on Integration of Firefly Algorithm and Quadratic Polynomial Equation”. In: *Intelligent Systems and Networks*. The International Conference on Intelligent Systems & Networks. ISSN: 2367-3389. Springer, Singapore, 2021, pp. 538–547. ISBN: 978-981-16-2094-2. DOI: 10.1007/978-981-16-2094-2_64. URL: https://link.springer.com/chapter/10.1007/978-981-16-2094-2_64 (visited on 05/18/2025).

[17] Eric V. Denardo. *Dynamic Programming: Models and Applications*. Courier Corporation, Dec. 27, 2012. 240 pp. ISBN: 978-0-486-15085-7.

[18] Wonteak Lim et al. “Hierarchical Trajectory Planning of an Autonomous Car Based on the Integration of a Sampling and an Optimization Method”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.2 (Feb. 2018), pp. 613–626. ISSN: 1558-0016. DOI: 10.1109/TITS.2017.2756099. URL: https://ieeexplore.ieee.org/document/8242694 (visited on 05/18/2025).

[19] Wonteak Lim et al. “Hybrid Trajectory Planning for Autonomous Driving in On-Road Dynamic Scenarios”. In: *IEEE Transactions on Intelligent Transportation Systems* 22.1 (Jan. 2021), pp. 341–355. ISSN: 1558-0016. DOI: 10.1109/TITS.2019.2957797. URL: https://ieeexplore.ieee.org/abstract/document/8932661 (visited on 05/18/2025).

[20] Joshue Perez Rastelli, Ray Lattarulo, and Fawzi Nashashibi. "Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 510–515. ISBN: 1-4799-3638-3.

[21] Hiroshi Fuji et al. "Trajectory planning for automated parking using multi-resolution state roadmap considering non-holonomic constraints". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. 2014 IEEE Intelligent Vehicles Symposium. ISSN: 1931-0587. June 2014, pp. 407–413. DOI: 10.1109/IVS.2014.6856433. URL: https://ieeexplore.ieee.org/document/6856433 (visited on 05/18/2025).

[22] Carl Bergenhem et al. "Overview of platooning systems". In: *Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012)*. 2012.

[23] Plamen Petrov and Fawzi Nashashibi. "Modeling and nonlinear adaptive control for autonomous vehicle overtaking". In: *IEEE Transactions on Intelligent Transportation Systems* 15.4 (2014). ISBN: 1524-9050 Publisher: IEEE, pp. 1643–1656.

[24] Xuemin Hu et al. "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles". In: *Mechanical Systems and Signal Processing* 100 (Feb. 1, 2018), pp. 482–500. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2017.07.019. URL: https://www.sciencedirect.com/science/article/pii/S0888327017303825 (visited on 05/18/2025).

[25] Gregor Klančar et al. "Drivable path planning using hybrid search algorithm based on E* and Bernstein–Bézier motion primitives". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51.8 (2019). ISBN: 2168-2216 Publisher: IEEE, pp. 4868–4882.

[26] Changxi You et al. "Autonomous planning and control for intelligent vehicles in traffic". In: *IEEE Transactions on Intelligent Transportation Systems* 21.6 (2019). ISBN: 1524-9050 Publisher: IEEE, pp. 2339–2349.

[27] A. Chebly, R. Talj, and A. Charara. "Coupled Longitudinal and Lateral Control for an Autonomous Vehicle Dynamics Modeled Using a Robotics Formalism". In: *IFAC-PapersOnLine* 50.1 (July 2017), pp. 12526–12532. ISSN: 24058963. DOI: 10.1016/j.ifacol.2017.08.2190. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896317328604 (visited on 04/25/2025).

[28] Ali Jnadi, Karam Almaghout, and Alexander Pronin. "Performance Evaluation of PID, Stanley, and Hybrid (PID with Stanley) Control Algorithms on Bitum Line-Follower Road Coaster". In: *2024 International Ural Conference on Electrical Power Engineering (UralCon)*. 2024 International Ural Conference on Electrical Power Engineering (UralCon). Magnitogorsk, Russian Federation: IEEE, Sept. 27, 2024, pp. 770–774. ISBN: 979-8-3503-8602-8. DOI: 10.1109/UralCon62137.2024.10718709. URL: https://ieeexplore.ieee.org/document/10718709/ (visited on 04/25/2025).

[29] Adam Domina and Viktor Tihanyi. "Comparison of path following controllers for autonomous vehicles". In: *2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. 2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI). Herlany, Slovakia: IEEE, Jan. 2019, pp. 147–152. ISBN: 978-1-7281-0250-4. DOI: 10.1109/SAMI.2019.8782719. URL: https://ieeexplore.ieee.org/document/8782719/ (visited on 04/25/2025).

[30] Wael Farag. “Complex Trajectory Tracking Using PID Control for Autonomous Driving”. In: *International Journal of Intelligent Transportation Systems Research* 18.2 (May 2020), pp. 356–366. ISSN: 1348-8503, 1868-8659. DOI: 10.1007/s13177-019-00204-2. URL: http://link.springer.com/10.1007/s13177-019-00204-2 (visited on 04/14/2025).

[31] Lei Li, Jun Li, and Shiyi Zhang. “Review article: State-of-the-art trajectory tracking of autonomous vehicles”. In: *Mechanical Sciences* 12.1 (Apr. 16, 2021), pp. 419–432. ISSN: 2191-916X. DOI: 10.5194/ms-12-419-2021. URL: https://ms.copernicus.org/articles/12/419/2021/ (visited on 04/10/2025).

[32] Qiangqiang Yao et al. “Control Strategies on Path Tracking for Autonomous Vehicle: State of the Art and Future Challenges”. In: *IEEE Access* 8 (2020), pp. 161211–161222. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3020075. URL: https://ieeexplore.ieee.org/document/9179748/ (visited on 04/26/2025).

[33] R. Craig Coulter. *Implementation of the Pure Pursuit Path Tracking Algorithm*. 1992. URL: https://www.ri.cmu.edu/pub_files/pub3/coulter_r_craig_1992_1/coulter_r_craig_1992_1.pdf (visited on 04/25/2025).

[34] Chinmay Vilas Samak, Tanmay Vilas Samak, and Sivanathan Kandhasamy. “Control Strategies for Autonomous Vehicles”. In: Lentin Joseph and Amit Kumar Mondal. *Autonomous Driving and Advanced Driver-Assistance Systems (ADAS)*. 1st ed. Boca Raton: CRC Press, Nov. 12, 2021, pp. 37–86. ISBN: 978-1-003-04838-1. DOI: 10.1201/9781003048381-3. URL: https://www.taylorfrancis.com/books/9781003048381/chapters/10.1201/9781003048381-3 (visited on 04/12/2025).

[35] Jarrod M Snider. “Automatic Steering Methods for Autonomous Automobile Path Tracking”. In: (Feb. 2009).

[36] Brian Paden et al. “A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles”. In: *IEEE Transactions on Intelligent Vehicles* 1.1 (Mar. 2016), pp. 33–55. ISSN: 2379-8904, 2379-8858. DOI: 10.1109/TIV.2016.2578706. URL: http://ieeexplore.ieee.org/document/7490340/ (visited on 04/25/2025).

[37] Gabriel M. Hoffmann et al. “Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing”. In: *2007 American Control Conference*. 2007 American Control Conference. ISSN: 0743-1619. New York, NY, USA: IEEE, July 2007, pp. 2296–2301. ISBN: 978-1-4244-0988-4 978-1-4244-0989-1. DOI: 10.1109/ACC.2007.4282788. URL: http://ieeexplore.ieee.org/document/4282788/ (visited on 05/10/2025).

[38] Jun Yang et al. “An Algorithm of Curved Path Tracking with Prediction Model for Autonomous Vehicle”. In: *2017 13th International Conference on Computational Intelligence and Security (CIS)*. 2017 13th International Conference on Computational Intelligence and Security (CIS). Hong Kong: IEEE, Dec. 2017, pp. 405–408. ISBN: 978-1-5386-4822-3. DOI: 10.1109/CIS.2017.00094. URL: http://ieeexplore.ieee.org/document/8288516/ (visited on 05/20/2025).

[39] Hassan Talebi Abatari and Abdolreza Dehghani Tafti. “Using a fuzzy PID controller for the path following of a car-like mobile robot”. In: *2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*. 2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM 2013). Tehran: IEEE, Feb. 2013, pp. 189–193. ISBN: 978-1-4673-5811-8 978-1-4673-5809-5 978-1-4673-5810-1. DOI: 10.1109/ICRoM.2013.6510103. URL: http://ieeexplore.ieee.org/document/6510103/ (visited on 05/20/2025).

[40] Gaining Han et al. “The Lateral Tracking Control for the Intelligent Vehicle Based on Adaptive PID Neural Network”. In: *Sensors* 17.6 (May 30, 2017), p. 1244. ISSN: 1424-8220. DOI: 10.3390/s17061244. URL: https://www.mdpi.com/1424-8220/17/6/1244 (visited on 05/20/2025).

[41] Ying Bai and Dali Wang. “Fundamentals of Fuzzy Logic Control — Fuzzy Sets, Fuzzy Rules and Defuzzifications”. In: *Advanced Fuzzy Logic Technologies in Industrial Applications*. Ed. by Ying Bai, Hanqi Zhuang, and Dali Wang. Series Title: Advances in Industrial Control. London: Springer London, 2006, pp. 17–36. ISBN: 978-1-84628-468-7. DOI: 10.1007/978-1-84628-469-4_2. URL: http://link.springer.com/10.1007/978-1-84628-469-4_2 (visited on 05/02/2025).

[42] Xinyu Wang et al. “Lateral control of autonomous vehicles based on fuzzy logic”. In: *2013 25th Chinese Control and Decision Conference (CCDC)*. 2013 25th Chinese Control and Decision Conference (CCDC). Guiyang, China: IEEE, May 2013, pp. 237–242. ISBN: 978-1-4673-5534-6 978-1-4673-5533-9 978-1-4673-5532-2. DOI: 10.1109/CCDC.2013.6560927. URL: http://ieeexplore.ieee.org/document/6560927/ (visited on 04/25/2025).

[43] Sami Allou, Youcef Zennir, and Aissa Belmeguenai. “Fuzzy logic controller for autonomous vehicle path tracking”. In: *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. 2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA). Monastir: IEEE, Dec. 2017, pp. 328–333. ISBN: 978-1-5386-1084-8. DOI: 10.1109/STA.2017.8314969. URL: http://ieeexplore.ieee.org/document/8314969/ (visited on 05/20/2025).

[44] J. Ackermann et al. “Linear and nonlinear controller design for robust automatic steering”. In: *IEEE Transactions on Control Systems Technology* 3.1 (Mar. 1995), pp. 132–143. ISSN: 10636536. DOI: 10.1109/87.370719. URL: http://ieeexplore.ieee.org/document/370719/ (visited on 05/21/2025).

[45] Gilles Tagne, Reine Talj, and Ali Charara. “Immersion and invariance vs sliding mode control for reference trajectory tracking of autonomous vehicles”. In: *2014 European Control Conference (ECC)*. 2014 European Control Conference (ECC). Strasbourg, France: IEEE, June 2014, pp. 2888–2893. ISBN: 978-3-9524269-1-3. DOI: 10.1109/ECC.2014.6862436. URL: http://ieeexplore.ieee.org/document/6862436/ (visited on 05/21/2025).

[46] Jinghua Guo, Yugong Luo, and Keqiang Li. “An Adaptive Hierarchical Trajectory Following Control Approach of Autonomous Four-Wheel Independent Drive Electric Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.8 (Aug. 2018), pp. 2482–2492. ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2017.2749416. URL: https://ieeexplore.ieee.org/document/8057584/ (visited on 05/21/2025).

[47] Chuan Hu et al. “MME-EKF-Based Path-Tracking Control of Autonomous Vehicles Considering Input Saturation”. In: *IEEE Transactions on Vehicular Technology* 68.6 (June 2019), pp. 5246–5259. ISSN: 0018-9545, 1939-9359. DOI: 10.1109/TVT.2019.2907696. URL: https://ieeexplore.ieee.org/document/8675462/ (visited on 05/21/2025).

[48] Carlos E. García, David M. Prett, and Manfred Morari. “Model predictive control: Theory and practice—A survey”. In: *Automatica* 25.3 (May 1989), pp. 335–348. ISSN: 00051098. DOI: 10.1016/0005-1098(89)90002-2. URL: https://linkinghub.elsevier.com/retrieve/pii/0005109889900022 (visited on 04/27/2025).

A Concept Evaluation Matrices

This appendix contains the evaluation matrices for each component within the SOTA analysis, chapter 2, where each component's method was evaluated. Each matrix includes the criteria, ratings, and total score. The rating scale was from 1 to 5, where higher values generally reflect better performance or lower cost, depending on the context. The evaluation matrices are a variant of the concept-scoring matrix method, but in this case, each criterion had equal weight, resulting in a maximum possible score for each method equal to the number of criteria times five. The ratings are based on the findings and insights gathered during the SOTA analysis. Explanations for each criterion can be seen in Table A.1.

Table A.1: Explanations for the criteria used

Criteria	**Explanation**
Monetary cost	The cost of the method
Adaptability	How easy the method can be adapted to different conditions
Consistency	How consistently the method performs across repeated trials under the same conditions.
Reliability	How reliably the method performs over time and under varying or challenging conditions.
Accuracy	How accurately the method performs, including position accuracy in simulation
Resolution	The resolution provided by the method
Robustness	How well the method can handle errors and maintain stability in real-world, unpredictable conditions
Path optimization	How well the method optimizes the path
High-speed capability	Whether the method can operate effectively at high speeds (e.g., above 2 m/s)
Computational simplicity	How simple the method is to compute, in terms of number and complexity of required operations.
Implementation simplicity	How simple the method is to implement or integrate using available tools and solutions. Also how familiar is the team with this method

A.1 Perception

Table A.2 shows the evaluation matrix for perception methods.

Table A.2: Evaluation matrix for perception methods

Criteria	**LiDAR**	**Stereo camera**	**Mono camera**	**LPS**	**Radar**	**Ultrasonic**
Monetary Cost	3	2	4	3	4	5
Adaptability	5	5	5	5	5	5
Consistency	5	3	2	5	2	1
Accuracy	5	4	3	5	2	1
Resolution	5	5	5	4	2	1
Computational Simplicity	3	1	3	1	1	1
Implementation simplicity	3	2	2	4	3	3
Total score (max 35)	**29**	**22**	**24**	**27**	**19**	**17**

A.2 Localization

Table A.3 shows the evaluation matrix for localization methods.

Table A.3: Evaluation matrix for localization methods

Criteria	**SLAM**	**UWB**	**Odometry**	**Vision**	**PF**	**Map-based**
Adaptability	4	3	3	5	5	2
Consistency	4	4	2	3	4	5
Accuracy	4	4	2	5	5	5
Computational simplicity	2	3	5	1	2	4
Implementation simplicity	3	3	5	3	2	4
Total score (max 25)	**17**	**17**	**17**	**17**	**18**	**20**

A.3 Path planning

Table A.4 shows the evaluation matrix for path planning methods.

Table A.4: Evaluation matrix for path planning methods

Criteria	**Graph Based**	**Sampling Based**	**Gradient Based**	**Optimization Based**	**Interpolation Curve**
Adaptability	3	4	4	3	3
Consistency	5	1	5	5	3
Path optimization	3	1	2	5	3
Computational simplicity	3	5	3	1	5
Implementation simplicity	5	4	3	1	4
Total score (max 25)	**19**	**15**	**17**	**15**	**18**

A.4 Path following

Table A.5 shows the evaluation matrix for path following methods.

Table A.5: Evaluation matrix for path following methods

Criteria	**Pure pursuit**	**Stanley**	**PID**	**Fuzzy**	**SMC**	**MPC**
Accuracy	4	5	3	3	4	5
Robustness	2	3	3	4	4	5
Reliability	5	5	4	3	2	5
High-speed capability (>2m/s)	4	4	3	3	4	5
Computational simplicity	5	5	4	3	4	2
Implementation simplicity	5	5	5	3	2	1
Total score (max 30)	**25**	**27**	**22**	**19**	**20**	**23**

B Risk Analysis

The appendix contains the Likelihood-Severity Matrix shown in table B.1 and the Risk Analysis Matrix, can be seen in table B.2, used to evaluate the potential risks during the development of the autonomous RC-car.

Table B.1: Likelihood-severity matrix

		Severity				
		1 (Negligible)	**2 (Minor)**	**3 (Moderate)**	**4 (Significant)**	**5 (Severe)**
Likelihood	**1 (Very rare)**	1 Safe	2 Safe	3 Safe	4 Bad	5 Bad
	2 (Unlikely)	2 Safe	4 Safe	6 Bad	8 Bad	10 Very bad
	3 (Probable)	3 Safe	6 Bad	9 Very bad	12 Very bad	15 Unacceptable
	4 (Likely)	4 Bad	8 Bad	12 Very bad	16 Unacceptable	20 Unacceptable
	5 (Almost certain)	5 Bad	10 Very bad	15 Unacceptable	20 Unacceptable	25 Unacceptable

Table B.2: Risk analysis

Source of risk	Risk description	Probable causes	Consequences	Preventative measures	Reactive measures	Likelyhood	Severity	Priority
Testing area (public parking lot)	Team member getting hit by the car during testing	Team member goes inside the testing area or car malfunctions	Team member leg injuries, damage to car, possible delays in project due to injured team member	Stay outside of designated testing area, preferably behind a physical barrier	Attempt to take control of the car with the remote controller	1	4	4
Testing area (public parking lot)	Civilian car gets hit by the RC car	Civilian entered testing area without team members noticing	Leg injuries, damage to car	Team members stay vigilant, stop all testing if civilian approaches testing site	Stop the car	2	5	10
Testing area (public parking lot)	Component overheats, causing fire in the car	Poor construction of car or the car was driven too close to it's limits	Loss of components, loss of control of car, fire may spread	Know the limits of the car, construct it properly	Use a fire extinguisher	1	3	3
Testing area (public parking lot)	Wheel or other moving part comes off	Poor construction of the car, lack of maintenance	Part may break, part may hit a person, car might crash	Make sure everything is properly tightened and perform regular maintenance checks	Stop the car	2	3	6
Team	Ordered components arrive late	Unreliability in shipping companies	Project delays	Plan ahead, order parts early	Check if KTH has temporary spares available until new components arrive	4	2	8
Team	Components do not work properly	Damage during transport, during use or components were always defective	Car may lose control, new parts need to be ordered, project delays	Test all components, perform regular maintenance, order spare components	Stop the car, change defective components	3	3	9
Team	Team member(s) absent over long periods of time, or uneven work distribution	Poor communication and/or planning	Project delays, internal conflict, low team morale	Be transparent and honest, communicate and plan properly	Review current plan, reach an agreement on responsibilities with absent member(s)	3	4	12
Team	Low team morale	Struggles in making the project work, loss of interest in project, internal conflicts in team	Project delays, risk of more conflicts, downward spiral	Communicate and plan properly, keep doing fun teambuilding/hangout activites to keep friendliness and morale high	Do teambuilding/ hangout activities, help each other out	4	5	20
Team	Integration Hell	Poor planning, mistakes in designs, oversights	Project delays	Be communicative, work together with integration in mind from the start	Review current plan, rethink feasibility, change required design(s)	5	4	20
Team	CanEduDev unavailable to send parts or discuss issues	Poor communication with stakeholders	Project delays	Keep having thorough communication with stakeholders, be transparent	Try to find a solution with available teaching team at KTH	2	2	4
Team	Car component breaks/malfunctions during time-sensitive testing	Damage during transport, during use or components were always defective	Project delays until a new time slot for that testing opens up	Test all components, perform regular maintenance, order spare components	Stop the car, change defective components	2	3	6

C Fall Time Plan

This appendix provides a detailed overview of the fall plan.

Phase 1: Development (Sep 1 – Sep 18)

- Finalize vehicle platform and architecture
- Mount sensors (IMU, encoders, LiDAR)
- Develop basic simulation environment (Simulink, Gazebo)
- Set up ROS
- Start collecting sample sensor data
- Begin report draft (project scope, system architecture)

Phase 2: Subsystem Testing (Sep 18 – Oct 6)

- Implement and tune low-level control (PID for steering/speed)
- Test individual modules (localization, control) in simulation and hardware
- Compare real-world and simulated data for calibration
- Conduct small-scale driving tests
- Write intermediate report section: modeling, testing results

Phase 3: Integration (Oct 6 – Nov 6)

- Integrate localization and control systems
- Connect software and hardware (ROS)
- Test integrated system on vehicle (simulation + track)
- Update report with integration results

Phase 4: Evaluation and Optimization (Nov 6 – Dec 8)

- Tune system based on full-track testing
- Improve robustness (noise handling, edge cases)
- Final safety validation
- Run multiple full test runs
- Finalize project report and presentation

Administrative and Deliverable Milestones

- **Aug 25:** Initial meeting and revisit role assignment
- **Sep 18:** Component tested and finished modelling
- **Oct 6:** Proof of concept by individual modules validated
- **Nov 6:** Integrated system and intermediate report
- **Dec 8:** Submit final report and live working demo
- Weekly: Internal updates and shared documentation