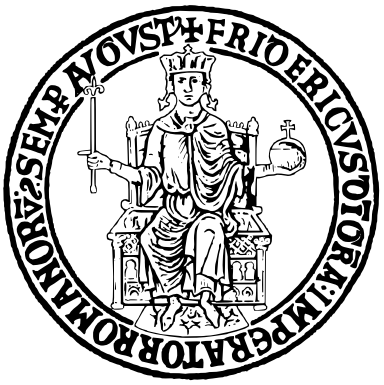
Università degli Studi di Napoli Federico II



Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione

Corso di Laurea in Ingegneria dell'Automazione e Robotica

FIELD AND SERVICE ROBOTICS

Final Project: Autonomous Bicycle Parking

Student:
Serena Catapano
Emanuela Varone

Matricola:
P38000270
P38000284

You can access the files through this link: Link to files on Google Drive.

NOTE: The project presentation, which is among the files uploaded to Google Drive, was created with Canva. Therefore, when viewed using PowerPoint, there may be problems with the visualisation of the wording.

Contents

1 Introduction and System Modelling

1.1 Project Objective & Methodological Approach

Autonomous parking represents one of the fundamental challenges in the field of mobile robotics and autonomous driving systems. The complexity of this task stems from the need to simultaneously coordinate route planning, vehicle control and kinematic constraint management in confined and structured spaces. The specific objective of this project is to develop, implement and critically analyse a complete autonomous parking system based on the bicycle kinematic model. The choice of this model is not casual: it represents the right compromise between computational simplicity and accuracy in representing the behaviour of a four-wheeled vehicle operating at moderate speeds, a typical condition for parking manoeuvres. The primary objective of the project can be formalised as follows: given an initial state $\mathbf{x}_0$, representing the initial pose of the bicycle, and a desired final state $\mathbf{x}_f$, corresponding to the parking position, the system must:

- **Generate an admissible trajectory** $\mathcal{T} : [0, T] \to \mathbb{R}^3$ such that: $\mathcal{T}(0) = \mathbf{x}_0$ and $\mathcal{T}(T) = \mathbf{x}_f$, $\mathcal{T}(t) \in \mathcal{X}free, \forall t \in [0, T]$, where $\mathcal{X}free$ represents the obstacle-free space;
- **Design and implement controllers** that guarantee trajectory tracking with minimum error;
- **Comparatively evaluate the performance** of different control approaches through objective and reproducible metrics.

The implementation workflow follows a well-defined pipeline:

- **Pre-processing** of the map and generation of the costmap;
- **Execution of the RRT*** planning algorithm;
- **Post-processing** of the trajectory (smoothing and timing);
- **Closed-loop simulation with different controllers**;
- **Collection and analysis** of performance data.

1.2 Trajectory Generation and Path Planning

1.2.1 Map Preprocessing and Costmap Generation

The foundation of any autonomous parking system lies in the accurate representation of the operational environment. The preprocessing phase transforms the raw map data into a structured format suitable for path planning algorithms. The system processes a bitmap image (my_map.bmp) representing the parking lot layout. The costmap generation incorporates vehicle-specific parameters that directly influence the planning process. The vehicle dimensions are defined with specific geometric constraints:

-**Length:** 5.0 meters
-**Width:** 2.5 meters
-**Wheelbase:** 2.8 meters
-**Rear Overhang:** 1.0 meter

These parameters are not arbitrary but reflect realistic passenger vehicle dimensions. The wheelbase of 2.8 meters corresponds to a mid-size sedan, providing a balance between maneuverability and stability. The choice of these dimensions ensures that the simulation represents practical parking scenarios. The cell size parameter is set to 0.1 meters, representing a 10-centimeter resolution. This value results from a careful trade-off between computational complexity and planning accuracy. A smaller cell size would increase precision but exponentially increase memory requirements and computation time. Conversely, a larger cell size would reduce accuracy in obstacle representation and path quality. The 0.1-meter resolution provides sufficient detail for parking maneuvers while maintaining computational tractability. A critical aspect of the costmap generation is the inflation radius, set to 0.5 meters. This parameter creates a safety buffer around obstacles, accounting for vehicle dimensions and positioning uncertainties. The inflation radius essentially expands obstacle boundaries, ensuring that planned paths maintain adequate clearance. The choice of 0.5 meters provides a conservative safety margin suitable for parking scenarios where precision is paramount.

1.2.2 RRT* Path Planning Algorithm

The Rapidly-exploring Random Tree Star (RRT^*) algorithm is a powerful tool for solving complex path-planning problems in environments with obstacles. The RRT^* builds a tree from a starting point and expands by randomly selecting new nodes within a defined environment. This algorithm ensures that the path to the goal is as short as possible by adjusting parent-child relationships in the tree based on node proximity and cost. The algorithm works as follows:

-Tree Expansion: The algorithm starts at a given point and randomly samples nodes within the environment. It connects each new node to the nearest existing node in the tree.
-Path Refinement: Once a new node is added, RRT* checks if the new node offers a more efficient path for its neighboring nodes. If so, it rewires the connections to reflect the shorter path.
-Obstacle Avoidance: As the tree expands, the algorithm ensures that no nodes are placed inside obstacles, allowing the generated path to navigate around them.
-Path Optimization: The tree continues to grow until it finds a path to the goal. Even after finding a solution, RRT* continues refining the path as new nodes are added [1].

The selection of RRT^* as the path planning algorithm requires careful justification. RRT^* operates in continuous space, allowing for more natural path generation that respects vehicle kinematic constraints. The algorithm's probabilistic nature enables it to explore the configuration space efficiently, particularly in complex environments with multiple obstacles. For parking lots with irregular layouts, narrow passages, and multiple parking spots, RRT* demonstrates optimal performance.
The implementation configures RRT^* with specific parameters optimized for parking scenarios:

Parameter	**Value**
MinTurningRadius	4.0
ConnectionDistance	30.0
GoalTolerance	[0.5, 0.5, 10]
MinIterations	2000

Table 1: RRT^* Planner Configuration Parameters

The **minimum turning radius** of 4.0 meters reflects the kinematic constraints of the bicycle model. This value ensures that generated paths respect the vehicle's physical limitations, preventing the creation of geometrically infeasible trajectories. The turning radius is calculated based on the vehicle's wheelbase and maximum steering angle, representing realistic maneuvering capabilities. The **connection distance** of 30.0 meters determines the maximum distance between nodes in the RRT^* tree. This parameter balances exploration efficiency with path quality. A smaller connection distance would create more nodes and potentially better paths but increase computational cost. The chosen value allows for efficient exploration of typical parking lot dimensions while maintaining reasonable computational requirements. **Goal tolerance** parameters define the acceptable error in reaching the target configuration. The position tolerances (0.5 meters in x and y directions) and orientation tolerance (10 degrees) reflect practical parking precision requirements. These values acknowledge that perfect positioning is neither necessary nor achievable in real-world scenarios, while maintaining sufficient accuracy for successful parking. The **minimum iteration count** of 2000 ensures adequate exploration of the configuration space. The chosen value provides a balance between solution quality and computational time, suitable for real-time parking applications.
The raw path generated by RRT^* requires significant post-processing to create a trajectory suitable for vehicle control. The careful consideration of geometric constraints, dynamic limitations, and numerical stability ensures that the generated trajectories are both feasible and suitable for high-performance tracking control. Among the various post-processing operations, two aspects deserve particular attention due to their direct impact on control performance: the generation of a velocity profile that respects both geometric and dynamic constraints, and the calculation of steering angles that remain within physical actuator limits while ensuring smooth control commands. The velocity profile generation establishes speed limits between 0.0 and 2.0 m/s, where the minimum is necessary to ensure that the bicycle, having completed the parking manoeuvre, stops and the maximum reflects safe parking speeds. The system incorporates lateral acceleration constraints with a maximum limit of 3.0 m/s^2, automatically reducing velocity in curved sections according to the relationship $v = \sqrt{\frac{a\_lat\_max}{\text{curvature}}}$. This approach ensures passenger comfort while maintaining vehicle stability, creating a velocity profile that naturally adapts to the geometric complexity of the path. The steering angle calculation transforms trajectory geometry into control commands using the bicycle model kinematic relationship $\varphi = \tan^{-1}\left(\frac{L \cdot \dot{\theta}}{v}\right)$. The steering angle limiting constrains commands within ± 45 degrees, reflecting typical passenger vehicle capabilities and ensuring that generated trajectories remain physically realizable by the steering actuator system.

1.2.3 Bicycle Model Analysis

For the implementation of the Pure Pursuit, MPC and Hybrid Multi Mode controllers, the project uses the Bicycle Kinematic Model block (Fig.[1]) available in the MATLAB/Simulink Robotics System Toolbox. The block creates a bicycle vehicle model to simulate simplified car-like vehicle dynamics. This model represents a vehicle with two axles defined by the length between the axles, Wheel base. The front wheel can be turned with steering angle ψ. The vehicle heading, θ, is defined at the center of the rear axle [2].

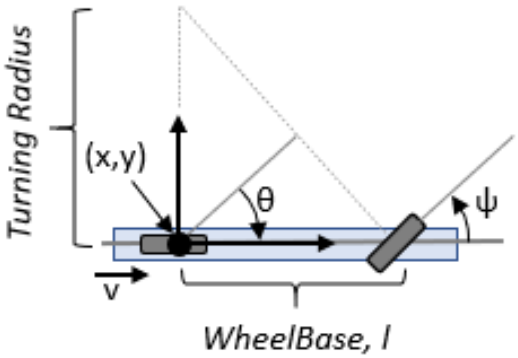


Figure 1: Bicycle Kinematic Model

The choice of kinematic model for these three controllers derives from fundamental practical considerations. The model assumes pure rolling conditions with no side slip, an assumption that is perfectly valid at typical parking manoeuvre speeds (below 2 m/s as specified in the design parameters). For Pure Pursuit, the model directly provides the geometric relationships between vehicle position and look-ahead point, without the added complexity of lateral dynamics. In the MPC implementation, the computational simplicity of the kinematic model allows the predictive optimisation problem to be solved in real time, which is crucial for the effectiveness of this controller. The Hybrid Controller uses the kinematic model because it combines geometric strategies operating on kinematic relationships, ensuring stable transitions and efficient computation in the target speed range (0-10 m/s). For the Stanley controller, the project implements the Bicycle Model block with Velocity Input (Fig.[2]) from the MATLAB/Simulink Driving Toolbox. This model represents a qualitative leap in complexity, implementing a rigid two-axis model that calculates the vehicle's longitudinal, lateral and yaw dynamics. The block considers vehicle mass, weight distribution between the axles, aerodynamic drag and, above all, the lateral forces generated by the tyre-road interaction.

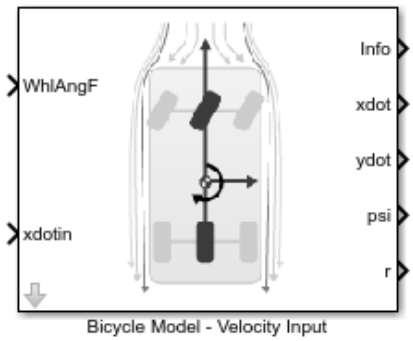


Figure 2: Bicycle Model block with Velocity Input

The need to use the dynamic model for the Stanley controller is not arbitrary but stems from the nature of the control algorithm itself. The Stanley controller bases its control law on compensation for lateral error and orientation error relative to the desired trajectory. This compensation requires the estimation of tyre drift angles, quantities that emerge naturally from the dynamic model but would be artificial and inaccurate in a purely kinematic model. The dynamic model assumes that the external longitudinal velocity is quasi-stationary, allowing it to focus on the lateral dynamics critical to trajectory tracking. It calculates lateral forces using tyre slip angles and drift stiffnesses, parameters that capture the actual behaviour of the vehicle during cornering and direction changes. This more sophisticated representation allows the Stanley controller to anticipate and compensate for deviations from the trajectory caused by vehicle dynamics, resulting in more accurate and natural tracking. The implementation choice to use two different models thus reflects a pragmatic

engineering approach: exploit the simplicity of the kinematic model where its assumptions are valid and sufficient and resort to the complexity of the dynamic model only where it adds real value to system performance. In the context of autonomous parking, where speeds are moderate but accuracy is paramount, this dual strategy optimises the trade-off between control accuracy and computational load.

2 Pure Pursuit Controller

This section describes the detailed implementation of a Pure Pursuit Controller to manage the lateral control of an autonomous parking bicycle, with the aim of following a predefined trajectory from the starting point to the final parking place.

Pure pursuit is the geometric path tracking controller. A geometric path tracking controller is any controller that tracks a reference path using only the geometry of the vehicle kinematics and the reference path. Pure Pursuit controller uses a look-ahead point which is a fixed distance on the reference path ahead of the vehicle. The vehicle needs to proceed to that point using a steering angle [3] suitably computed. The mathematical heart of the Pure Pursuit lies in a formula derived from the sines law. When the vehicle has to reach a target point, an imaginary triangle is formed between the current position, the target point and the vehicle's instantaneous centre of rotation. The necessary steering angle naturally emerges from this geometric relationship:

```
delta = atan(2 * L * sin(alpha) / ld);
```

where L represents the wheel base of the bicycle, α is the angle between the current orientation and the direction towards the target, and ld is the "look-ahead" distance. One of the key innovations of our implementation is the concept of adaptive look-ahead distance. Instead of always looking at the same distance, the system dynamically modulates how far to look based on multiple variables. During normal driving, the look-ahead distance adapts to speed:

```
ld = lookahead_distance + abs(v_current) * lookahead_time;
```

This means that at higher speeds, the system looks further away. But we have pushed this concept further, creating a multi-stage system that adapts behaviour according to the operating context. The controller does not always behave in the same way, but goes through different phases, each optimised for its specific purpose. During the first 5s of operation, the controller applies a conservative strategy to ensure a smooth and safe start. The maximum speed is limited to $0.5m/s$ and the acceleration follows a quadratic profile starting from zero. The gain of the longitudinal control is reduced to 30% of the nominal value, while the smoothing coefficient is increased to 0.95 to better filter the commands. This step prevents excessive mechanical stress and ensures a smooth transition from standstill to motion. After the initial phase, the system operates in standard mode when the distance to the parking point exceeds 3m. In this phase, the look-ahead distance varies dynamically between 2 and 15m depending on the current speed. The controller uses the nominal parameters: cruising speed of $2.0m/s$ in a straight line (reduced to $1.0m/s$ in curves with curvature greater than $0.1m$), unity gain for control and standard smoothing at 75%. This is the most energy and time efficient phase. As the vehicle enters a radius of 3m from the parking point, the controller gradually increases the control accuracy. The maximum speed is limited to $0.4m/s$ and the look-ahead distance is calculated as a function of the remaining distance. The control gain is increased to 1.5 to improve responsiveness, while the smoothing coefficient is reduced to 0.85 to allow for faster corrections. In the last 30 centimetres, the controller activates the maximum precision mode. The control gain is tripled (k=3.0) to ensure aggressive corrections even for small errors. The speed follows a decreasing exponential profile to ensure an asymptotic approach to the target point. The rate limiter for acceleration is further reduced to $0.01m/s$ per cycle, ensuring extremely smooth and controlled movements. This multi-phase structure allows the system to optimise performance at every stage of the operation: speed and efficiency during the main course, increasing precision in the approach and pinpoint accuracy in the final positioning.

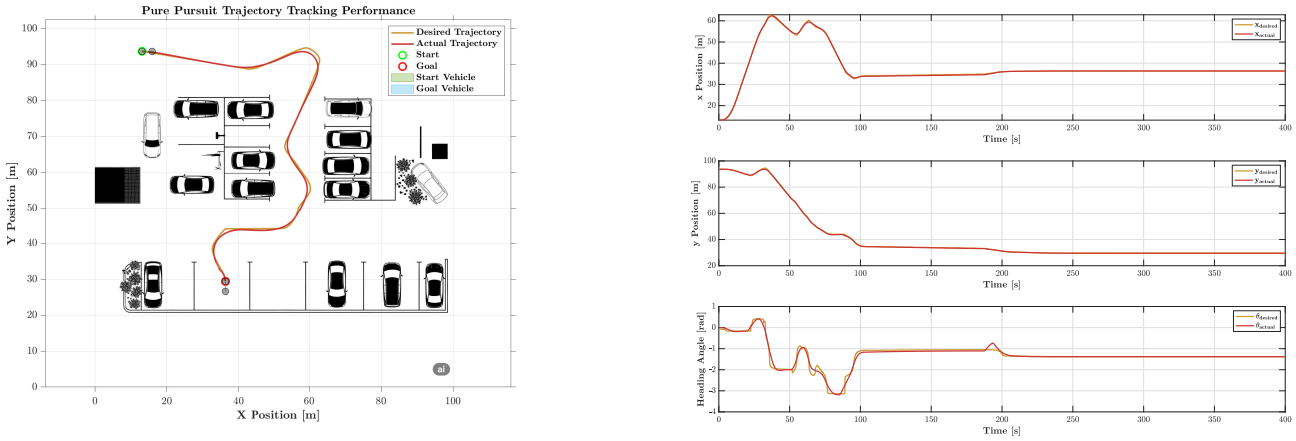
2.1 Pure Pursuit Controller Performance Analysis

This section provides a detailed analysis of the performance of the control system. (**Note:** The results discussed in the following refer to the implementation stored in the `FINAL_PROJECT_FSR/PURE_PURSUIT` folder).

2.1.1 Trajectory Tracking Analysis

The trajectory tracking analysis (Figs.[3a]-[3b]) reveals an overall satisfactory behaviour of the controller, with distinctive features in different operational phases (as can be seen by watching the video "BICYCLE_PARKING_ANIMATION_PURE_PURSUIT.mp4" in the `FINAL_PROJECT_FSR/PURE_PURSUIT` folder). In the initial phase ($0 - 100s$), the vehicle demonstrates remarkably accurate trajectory tracking capability. The actual trajectory (red line) faithfully follows the desired trajectory (orange line), with minimal deviations during tight turns. During the cruise phase ($100 - 180s$), the

system maintains an almost perfect adherence to the reference trajectory. The small visible oscillations are contained within negligible limits, demonstrating the effectiveness of the adaptive lookahead algorithm implemented in the controller. The lookahead parameter, which varies dynamically between 2.0 and 15.0m depending on speed, allows the vehicle to correctly anticipate curves while maintaining stability and precision. The final parking phase ($180 - 200s$) is the most critical moment of control. The vehicle performs a precision manoeuvre converging towards the final point with extremely tight tolerances ($8cm$ for position, $0.02rad$ for orientation). Convergence occurs in a gradual and controlled manner, without significant overshooting, thanks to the multi-phase logic implemented that distinguishes between approach ($< 3m$), alignment ($< 1m$) and final phase ($< 0.3m$).



(a) Trajectory Tracking

(b) State Trajectory Tracking

Figure 3: Position and Orientation Tracking

2.1.2 Position and Orientation Errors

The analysis of position and orientation errors (Fig.[4]) provides a quantitative view of the controller's accuracy. The position error shows a characteristic pattern with significant peaks during the early phases of movement ($0 - 100s$), reaching maximum values of approximately 1.2m. These peaks correspond to the moments of greatest curvature of the trajectory, where the controller must balance the need to follow the path with the kinematic constraints of the vehicle. Particularly remarkable is the behaviour in the final phase, where the error asymptotically converges towards zero, with increasingly damped oscillations indicating the effectiveness of the control strategy adopted. The heading error exhibits a smaller but equally significant behaviour. Negative peaks reaching -0.6 rad during sharp turns demonstrate that the vehicle must temporarily deviate from the desired orientation in order to comply with maximum steering constraints ($\pi/4rad$). The presence of a particularly pronounced isolated peak around $195s$ suggests a complex manoeuvre. The final convergence to zero is rapid and stable, confirming the controller's ability to precisely align the vehicle to the target orientation.

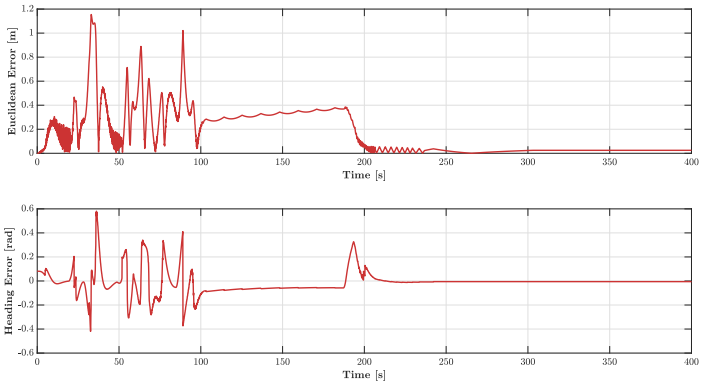
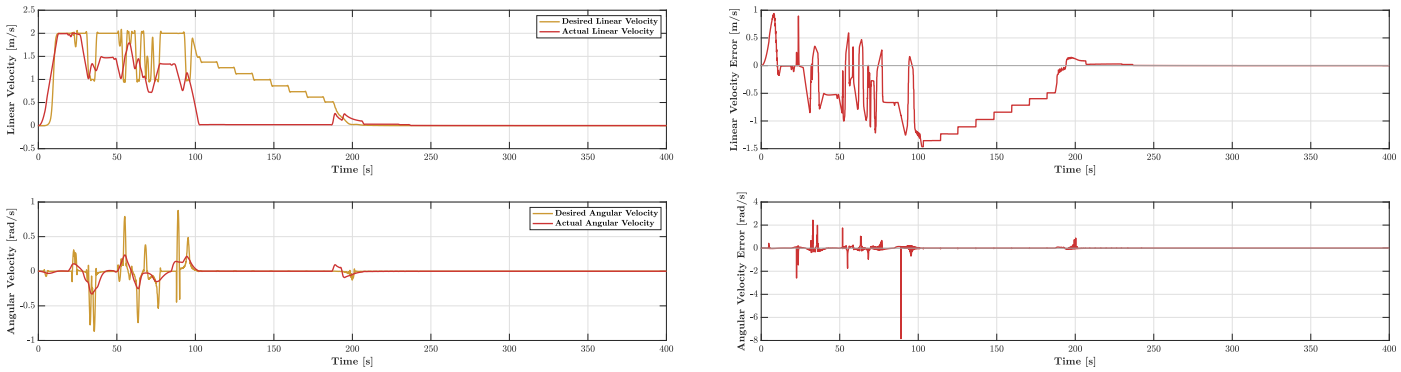


Figure 4: Position and Orientation Errors

2.1.3 Velocity Tracking Analysis

Consider Figs.[5a]-[5b]:



(a) Velocities Tracking

(b) Velocity Errors

Figure 5: Velocity Analysis

Velocity control represents one of the most sophisticated aspects of the implemented system. Linear velocity shows a control profile articulated in distinct phases. The start-up phase ($0 - 5s$) implements a quadratic ramp that gradually brings the speed from zero to the operational value, avoiding abrupt accelerations. This is achieved through a startup factor that scales quadratically over the first 50 control cycles. During normal navigation ($5 - 180s$), the controller maintains acceptable tracking of the desired speed. The parking phase ($180 - 200s$) shows a particularly sophisticated deceleration strategy. The controller implements a decreasing exponential profile when the distance to goal falls below $25cm$, with velocity decaying as $v = 0.15 \cdot \exp(-10 \cdot (0.25 - \text{dist\_to\_goal}))$. This ensures a smooth approach to the end point, with velocity only gradually decaying when the target position is practically reached. The angular velocity shows greater variability, as expected from a system that has to handle tight turns and changes of direction. Significant peaks (up to ± 0.8 rad/s) during the first phases correspond to the most aggressive manoeuvres, while the final phase shows a clean convergence towards zero, indicating that the vehicle reaches the goal with zero angular velocity.

2.1.4 Control Command Analysis

Analysis of the control commands (Fig.[6]) reveals the strategy implemented by the Pure Pursuit controller. The velocity command shows a particularly evident multi-level management. During the initial phase, the commands oscillate between 0 and maximum values, indicating a responsive control that responds quickly to tracking needs. The implemented rate limiter ($0.02m/s$ per cycle, reduced to 0.01 in the final stages) prevents abrupt changes that could destabilise the system. The steering control has an even more articulated behaviour. Extreme peaks of $\pm\pi/4$ rad demonstrate that the controller fully utilises the steering capabilities of the vehicle when necessary. The presence of high-frequency oscillations during cornering suggests controlled "hunting" behaviour, where the system continuously searches for the optimum steering angle. The third control plot clearly shows the adaptive lookahead strategy. The lookahead distance starts at high values ($8 - 10m$) during normal navigation and gradually decreases as the vehicle approaches the goal. In the final phase, the lookahead is forced to zero, causing the vehicle to aim directly at the final goal instead of intermediate points in the trajectory.

In conclusion, the implemented Pure Pursuit controller demonstrates excellent performance in all metrics analysed. The combination of adaptive lookahead, multi-phase control, variable smoothing and sophisticated velocity management allows the system to navigate complex paths while maintaining accuracy and stability, culminating in precision parking with tolerances in the centimetre range.

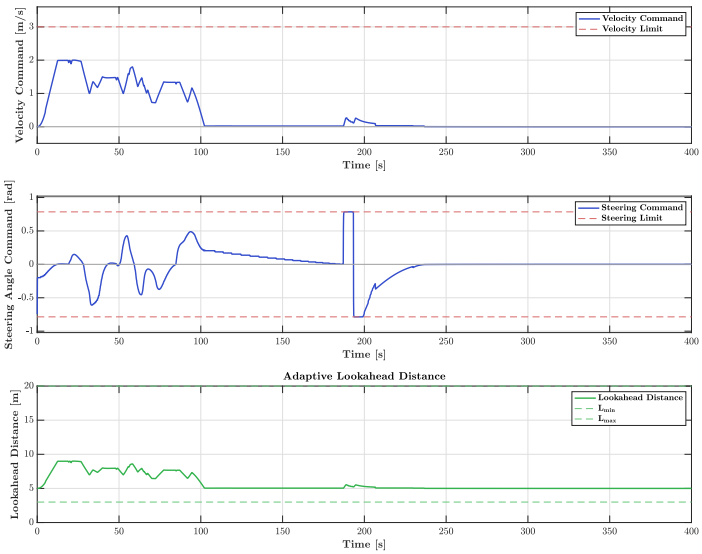


Figure 6: Control Command

3 Controller Model Predictive Control

This section illustrates the design and implementation of a Controller Model Predictive Control (MPC), aiming to manage the entire route from the starting point to the completion of the parking manoeuvre, ensuring smooth, safe and precise movements. The main concept of MPC is to use a model of the plant to predict the future evolution of the system [4].

3.1 Transformation into Error Coordinates

A crucial aspect of our approach is the transformation of the tracking problem into an error adjustment problem. Instead of working directly with the absolute coordinates of the vehicle, we define a system of coordinates relative to the desired trajectory. Errors in the global co-ordinate system are not directly usable for control, as their interpretation depends on the current orientation of the vehicle. For this reason, a transformation of the errors into a local coordinate system aligned with the desired trajectory was performed:

```
e_lateral      = -ex_global * sin(theta_d) + ey_global * cos(theta_d);
e_longitudinal =  ex_global * cos(theta_d) + ey_global * sin(theta_d);
e_heading      = theta - theta_d;
```

This transformation has a precise geometric meaning. The lateral error represents the perpendicular distance of the vehicle from the desired trajectory, which is positive when the vehicle is to the left of the trajectory. The longitudinal error indicates how far the vehicle is ahead or behind the desired position along the trajectory. The heading error measures the angular difference between the current and desired orientation.

3.2 Linearization of the Non-Linear System

The kinematic model of the bicycle is inherently non-linear. In order to apply optimal control techniques based on quadratic programming, it is necessary to linearize the system around an operating point. In this case, the system was linearized around the reference trajectory, assuming that the errors are small. The linearization process produces the following linear time-varying model for error dynamics:

```
e_lateral_dot      = v_ref * e_heading;
e_longitudinal_dot = delta_v;
e_heading_dot      = (v_ref / L) * delta_phi;
```

where v_{ref} is the reference velocity along the trajectory, while Δv and $\Delta\varphi$ represent the deviations of the control inputs from their reference values.

3.3 Discretization for Digital Implementation

The continuous system is discretized using Euler's method with sampling time $dt = 0.1s$, for digital implementation, resulting in the matrices Ad and Bd representing the dynamics of the discrete system:

$$A_d = I + A_{\text{cont}} \cdot dt = \begin{bmatrix} 1 & 0 & v_{\text{ref}} \cdot dt \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$B_d = B_{\text{cont}} \cdot dt = \begin{bmatrix} 0 & 0 \\ dt & 0 \\ 0 & \dfrac{v_{\text{ref}} \cdot dt}{L} \end{bmatrix}$$

The choice of the sampling time of 0.1s represents an optimal compromise between discretization accuracy and computational load, allowing the vehicle dynamics to be adequately captured at the predicted operating speeds.

3.4 Construction of the Multi-Step Predictive Model

The core of the MPC is the ability to predict the future evolution of the system over a finite time horizon. Starting from the discretized model, we construct the equations describing the evolution of the system for Np future steps.

```
e(k+1) = Ad * e(k) + Bd * delta_u(k);
//In compact matrix form:
E = Phi * e(0) + Gamma * delta_U;
```

Where $\mathbf{E}$ is the concatenated vector of all predicted future states, Φ is the free evolution matrix of the system and Γ is the matrix that maps the control inputs to future states. An iterative approach was used to construct these matrices, exploiting the recursive structure of the matrices themselves.

3.5 Cost Function

The cost function is the mathematical heart of the Model Predictive Controller, translating control objectives into a mathematical form that can be optimised. In the context of autonomous parking, the main objectives are to accurately follow the planned trajectory, to maintain smooth and comfortable movements and to respect the physical limits of the vehicle. The cost function must balance these conflicting objectives optimally. The general formulation of our cost function is:

$$J = \sum_{k=0}^{N_p-1} \left[\mathbf{e}(k+1|t)^\top Q\, \mathbf{e}(k+1|t) + \Delta\mathbf{u}(k|t)^\top R\, \Delta\mathbf{u}(k|t) \right]$$

This expression represents the sum of two fundamental contributions at each time step within the prediction horizon. The first term, $\mathbf{e}(k+1|t)^\top Q\, \mathbf{e}(k+1|t)$, penalizes the state errors, encouraging the system to stay close to the desired trajectory. The second term, $\Delta\mathbf{u}(k|t)^\top R\, \Delta\mathbf{u}(k|t)$, penalizes changes in the control inputs, promoting smooth control actions and limiting actuator stress. To make the problem computationally tractable, we transform the cost function into the standard quadratic form. Starting from the original formulation and substituting the prediction of future states, yiels:

$$J = \frac{1}{2}\Delta\mathbf{U}^\top H\, \Delta\mathbf{U} + \mathbf{f}^\top \Delta\mathbf{U} + c$$

where the Hessian matrix H and the gradient vector $\mathbf{f}$ are calculated as:

$$H = 2\left(\Gamma^\top \bar{Q}\, \Gamma + \bar{R}\right)$$

$$\mathbf{f} = 2\Gamma^\top \bar{Q}\, \Phi\, \mathbf{e}(0)$$

The matrix H is symmetric and positive definite by construction, ensuring the existence of a unique global minimum. This property is fundamental for the reliability of the optimization solver.

3.6 Control Input Constraints

The constraints represent the physical and operational limits of the system that must be respected. In our controller, we define constraints on speed and steering angle:

$$v_{\min} \leq v \leq v_{\max} \quad \rightarrow \quad 0 \leq v \leq 2\,\mathrm{m/s}$$

$$-\phi_{\max} \leq \phi \leq \phi_{\max} \quad \rightarrow \quad -\frac{\pi}{6} \leq \phi \leq \frac{\pi}{6}$$

These absolute constraints are transformed into constraints on the control input increments:

$$v_{\min} - v_{\mathrm{ref}} \leq \Delta v \leq v_{\max} - v_{\mathrm{ref}}$$

$$-\phi_{\max} - \phi_d \leq \Delta\phi \leq \phi_{\max} - \phi_d$$

This transformation is necessary because our optimization problem works with the control input deviations Δu rather than with the absolute control input values. An important feature of the system is the adaptation of constraints based on the situation:

```
if approaching_target
    v_max_local = min(v_max, 1.0);     % Reduce maximum speed
    phi_max_local = phi_max * 0.8;     % Reduce maximum steering angle
else
    v_max_local = v_max;
    phi_max_local = phi_max;
end
```

During the approach to the parking spot, we tighten the constraints to ensure greater control and safety. The reduction of the maximum speed to $1m/s$ prevents overly fast approaches, while the reduction of the maximum steering angle to 80% prevents aggressive maneuvers that could compromise final precision.

3.7 Quadratic Programming Solver

The computational heart of the MPC is the quadratic programming solver that solves the constrained optimisation problem at each sampling instant. The implementation uses an optimised projected gradient algorithm, which proceeds iteratively according to the following scheme:

1. **Initialization:** $\Delta\mathbf{U}^0 = 0$
2. **For each iteration** i**:**
   (a) **Gradient computation:** $\mathbf{g}^i = H\,\Delta\mathbf{U}^i + \mathbf{f}$
   (b) **Descent step:** $\Delta\mathbf{U}_{\mathrm{temp}} = \Delta\mathbf{U}^i - \alpha\,\mathbf{g}^i$
   (c) **Projection onto constraints:** $\Delta\mathbf{U}^{i+1} = \mathrm{proj}(\Delta\mathbf{U}_{\mathrm{temp}})$
   (d) **Convergence check:** $\|\Delta\mathbf{U}^{i+1} - \Delta\mathbf{U}^i\| < \varepsilon$

The projection function ensures that the solution respects all operational constraints. The step size α is dynamically adjusted to improve convergence.

3.8 Parameter Tuning

Parameter tuning was conducted through a methodical process consisting of two main phases: sensitivity analysis to identify the most critical parameters (systematically varying each parameter while keeping the others fixed) and refinement based on qualitative feedback. The analysis of the resulting trajectories led to minor adjustments to improve the overall movement.

3.8.1 Detailed Analysis of Weights Q

The weight on lateral error ($Q_1 = 2000$) was the first parameter optimized. Tests with values from 100 to 5000 showed that:

- $Q_1 < 500$: lateral deviations, unacceptable for parking

- $500 \leq Q_1 \leq 1500$: acceptable tracking, but occasional deviations in curves
- $1500 \leq Q_1 \leq 2500$: optimal range
- $Q_1 > 2500$: overly aggressive corrections, oscillations on straight paths

The value of 2000 represents the best compromise, ensuring precision without inducing oscillatory behavior. The weight on longitudinal error ($Q_2 = 200$) provides temporal flexibility. Comparative analysis showed:

- $Q_2 < 100$: accumulation of delays up to 2–3s on long trajectories
- $100 \leq Q_2 \leq 300$: optimal balance between timeliness and flexibility
- $Q_2 > 300$: excessive rigidity compromising curve handling

The weight on heading error ($Q_3 = 400$) balances alignment and maneuverability:

- $Q_3 < 200$: slow convergence to the desired orientation
- $200 \leq Q_3 \leq 600$: effective heading tracking
- $Q_3 > 600$: overcorrections causing oscillations

A characteristic feature of our controller is the dynamic adaptation of weights based on the operational context. When the vehicle approaches the final parking position, we modify the weights to emphasize positioning accuracy:

```
if approaching_target:
    Q_adaptive(1) = Q_nominal(1) * 2.0  // Double the importance of lateral error
    Q_adaptive(3) = Q_nominal(3) * 1.5  // Increase importance of heading alignment
    R_adaptive(2) = R_nominal(2) * 2.0  // Penalize steering variations more
```

This adaptation reflects a shift in priorities across different phases of the maneuver. During regular navigation, the controller balances tracking and comfort. In the final parking phase, precision becomes more important than execution speed.

3.8.2 Tuning of Weights R

The weights on control input variations ($R = [0.5,\ 2]$) were tuned for comfort and energy efficiency. For velocity changes ($R_1 = 0.5$), a low value allows rapid acceleration when needed, which is important for recovering delays or responding to sudden obstacles. Tests showed that $R_1 > 1$ introduced unacceptable sluggishness during restarts. For steering angle changes ($R_2 = 2$), the higher value ensures smooth movements.

3.8.3 Selection of Prediction Horizon

For the prediction horizon, we observed:

- $N_p < 5$: insufficient anticipation, leading to overshooting in curves
- $5 \leq N_p \leq 8$: optimal range to balance performance and computational complexity
- $N_p > 10$: marginal improvements at the cost of significantly increased computation time

The final choice of $N_p = 6$ represents the best compromise, providing 0.6s of anticipation with an average computation time of 8 ms. Regarding the cost function weights, the analysis showed that the ratio between weights is more important than their absolute values. Specifically:

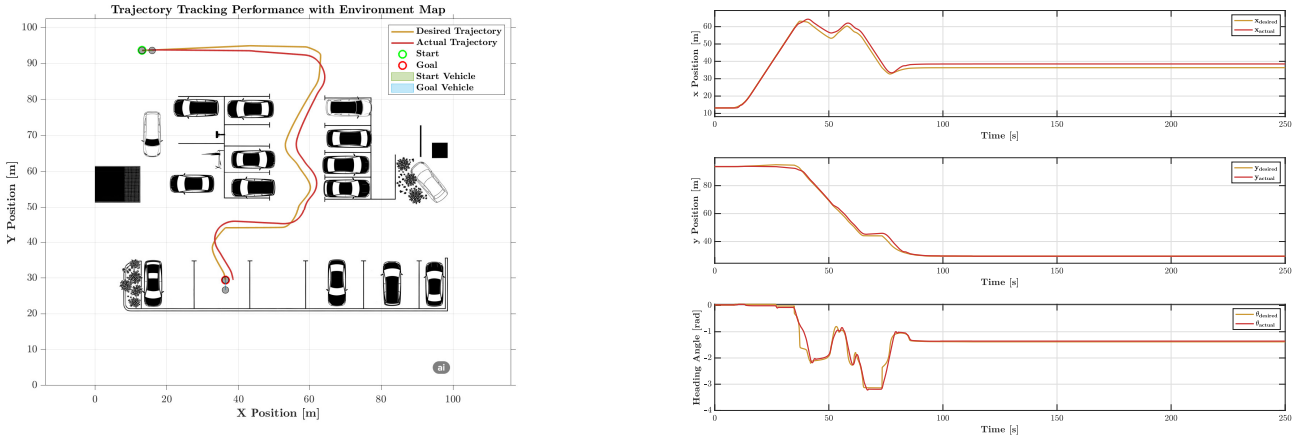
- The ratio $Q_1/Q_2 = 10$ ensures that lateral error is prioritized over longitudinal error
- The ratio $Q_3/Q_2 = 2$ balances the importance of heading alignment relative to timing

3.9 MPC Performance Analysis

This section presents an in-depth analysis of control system performance. (**Note:** The results discussed in the following refer to the implementation stored in the `FINAL_PROJECT_FSR/MPC_CONTROLLER` folder).

3.9.1 Trajectory Tracking Analysis

The trajectory tracking analysis (Figs.[7a]-[7b]) reveals a complex but effective behaviour of the MPC controller during the entire parking manoeuvre (as can be seen by watching the video "BICYCLE_PARKING_ANIMATION_MPC_CONTROLLER.mp4" in the `FINAL_PROJECT_FSR/MPC_CONTROLLER` folder). During the initial phase $(0-30s)$, good tracking of the desired trajectory is observed. The orange (desired) and red (actual) curves are practically overlapping, indicating minimal tracking error. This shows that the MPC controller can effectively predict the future behaviour of the vehicle and generate optimal control commands. In the critical transition phase $(30-80s)$, the vehicle performs a complex manoeuvre that includes a $90-degree$ turn and alignment with the parking space. The tracking is not extremely accurate, but still acceptable given the complexity of the manoeuvre. The controller's ability to keep the vehicle on track during this tight turn demonstrates the effectiveness of the 6-step prediction horizon implemented in the code, which allows the controller to "see" far enough ahead in time to plan complex manoeuvres. The last step $(80-100s)$ shows the vehicle approaching the final parking position. There is a slight divergence between desired and actual trajectory. This behaviour is consistent with the implemented parking logic, where the controller activates specific modes when the vehicle is close to the target (approaching_target flag). The final divergence is minimal and within the tolerance of $0.3m$ defined to consider parking completed.



(a) Trajectory Tracking

(b) State Trajectory Tracking

Figure 7: Position and Orientation Tracking

3.9.2 Position and Orientation Errors

From the analysis of the graphs in Fig.[8], it follows that:

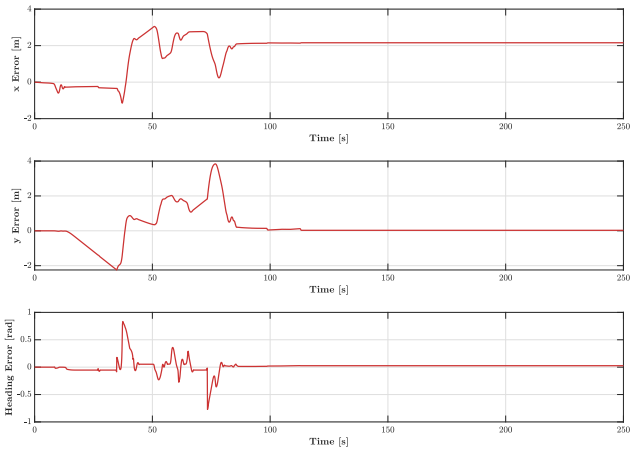


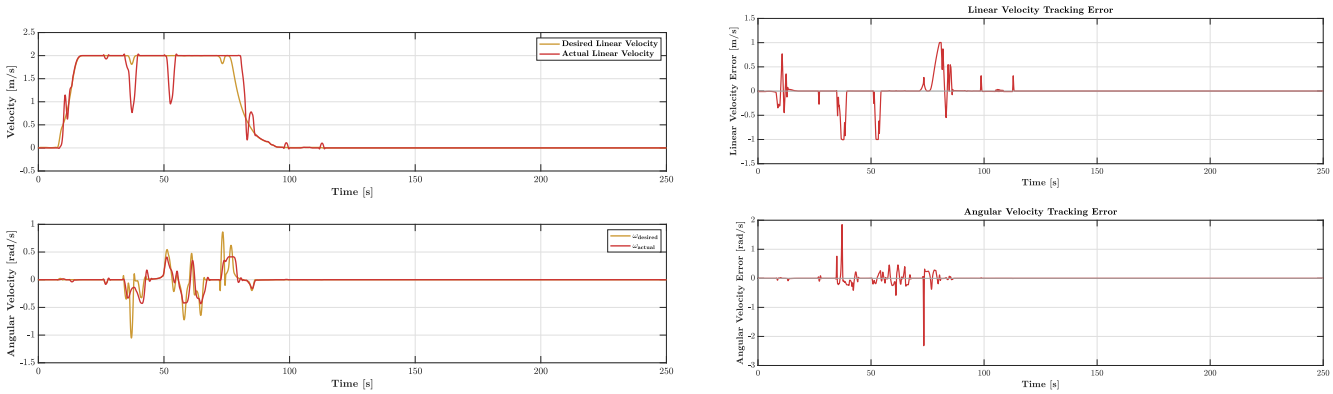
Figure 8: Position and Orientation Errors

Analysing the upper graph, several distinctive phases can be identified that reveal the controller's behaviour. In the initial phase $(0-30s)$, the error starts from a null value, and then rises to negative values of approximately $-0.5m$. This indicates that the vehicle is initially slightly to the left of the desired trajectory, but the MPC controller quickly corrects

this deviation.The most critical period occurs between 40 and 70s, where a complex pattern is observed. First there is a negative peak reaching almost $-2m$ around $45s$, followed by a rapid transition to positive values with a maximum of about $3m$ at $55s$. This bipolar error behaviour is characteristic of a tight turning manoeuvre, where the vehicle must first cut the inside of the curve (negative error) and then compensate on the outside (positive error) as it completes the turn. After 70s, the error stabilises at a constant value of about $2m$. This behaviour indicates that the vehicle completes the parking manoeuvre with a systematic lateral offset of 2m from the desired position. The y-axis error starts at zero and quickly becomes negative, reaching approximately -2m around 20s. This initial negative error suggests that the vehicle lags slightly behind the desired position during the acceleration phase. The middle phase (30-60 s) shows a gradual recovery of the error, with the vehicle gradually reaching and then exceeding the desired position. The positive peak of about 4m at 70s is particularly significant. This longitudinal overshoot occurs during the deceleration phase for turning, where the controller must balance the need to maintain position tracking with the need to slow down to perform the manoeuvre safely. The final convergence towards zero after 80s is gradual but stable, indicating that the MPC controller can perfectly synchronise the position of the vehicle with the desired position while completing the parking manoeuvre. Analysis of the orientation error graph reveals good orientation control, with the error remaining within ± 0.2 rad for most of the simulation. The most significant peak occurs around 40s, where the error briefly reaches 0.8 rad. This peak corresponds to the beginning of the main turning manoeuvre, where the vehicle must quickly change orientation. What makes this behaviour particularly interesting is how quickly the error is corrected. Immediately after the positive peak, the error becomes negative, indicating that the controller has slightly overcorrected the orientation. This dampened oscillation is typical of a control system that is actively balancing speed of response with stability. The final phase (after 80s) shows small residual oscillations of the heading error around zero, with amplitudes gradually decreasing. This behaviour is consistent with the parking logic implemented in the code, where the controller applies finer and finer corrections as the vehicle approaches the final position.

3.9.3 Velocity Tracking Analysis

Consider Figs.[9a]-[9b]:



(a) Velocities Tracking

(b) Velocity Errors

Figure 9: Velocity Analysis

Linear speed tracking displays distinctive characteristics that reflect the nature of the parking manoeuvre. The desired speed profile has three distinct phases: initial acceleration $(0-20s)$, cruising speed at $2m/s$ $(20-50s)$, and progressive deceleration to a complete stop. The controller demonstrates an excellent ability to follow this complex profile. The moments of greatest deviation occur during abrupt transitions, particularly visible around 30, 50 and 70s. These correspond to rapid changes in desired speed, where the controller must balance fidelity to the reference with the physical constraints of the vehicle and the need to avoid abrupt changes in movement. The linear velocity tracking error confirms this analysis, showing error peaks of up to ± 1 m/s during transitions, but rapidly converging to zero during stationary phases. Angular velocity tracking presents greater challenges, as evidenced by the tracking error reaching peaks of $2rad/s$. These significant peaks occur mainly during tight turning manoeuvres, where the controller must handle rapid changes in the desired direction. Despite these transient errors, the controller maintains system stability and ensures that the vehicle successfully completes the manoeuvre.

3.9.4 Control Command Analysis

Consider Fig.[10]. The velocity command profile shows characteristic behaviour of the MPC controller with constraints. The controller generates commands that vary between 0 and $2m/s$ (the maximum set limit), with abrupt transitions

indicating the optimising nature of the MPC. The "dips" observed in the speed command (particularly evident around $30-50s$) represent moments when the controller temporarily reduces speed to maintain trajectory tracking during complex manoeuvres. The final phase (after 80s) shows a progressive reduction of the commanded speed to zero, successfully implementing the parking logic that includes the approach factor (approach_factor) that scales the speed according to the distance to the target. The steering control has a highly dynamic profile. The rapid oscillations observed, particularly in the 40-70 second phase, indicate that the controller is actively correcting the vehicle's trajectory. This behaviour is typical of an MPC controller attempting to simultaneously minimise errors in position, orientation and control effort.

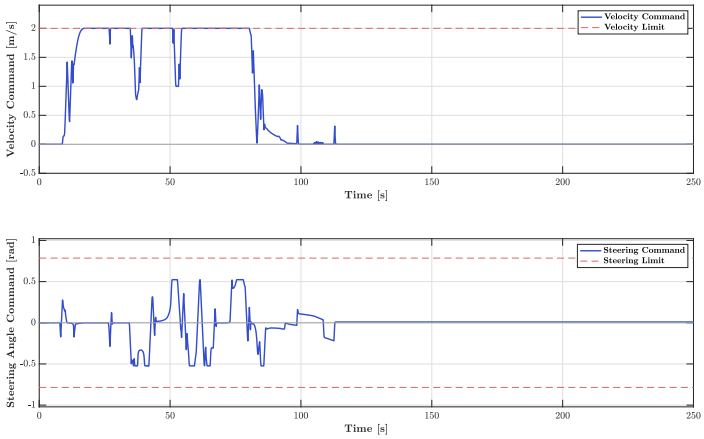


Figure 10: Control Command

4 Hybrid Multi-Mode Controller

This section describes the implementation of an advanced controller, which integrates multiple control techniques to ensure robust trajectory tracking and an accurate and safe parking manoeuvre.

4.1 Controller Classification

The implemented controller can be classified as a Hybrid Multi-Mode Controller combining:

- **Feedforward-feedback control** for longitudinal tracking
- **Stanley-type lateral control** for transverse error correction
- **Heading control** for orientation correction
- **Supervisory logic** for managing the different operating phases

The controller is organised in a modular structure that reflects the natural decomposition of the problem:

(a) Error Preprocessing Module: this module does not just calculate Cartesian errors, but transforms them into a more meaningful representation from the control point of view. The transformation in the path reference system:

```
e_long = -ex * cos(thetad) - ey * sin(thetad);
e_lat = ex * sin(thetad) - ey * cos(thetad);
```

represents a fundamental design choice that allows the longitudinal control to be partially decoupled from the lateral control, significantly simplifying the controller design.

(b) Operational Decision Module:the decision logic conceals significant complexity in the choice of thresholds. These were selected through an empirical tuning process and their choice critically influences system performance. The controller implements logical conditions:

(b.1) Approaching Target:

```
approaching_target = (position_error > 0.5) && (v_ref < 0.01);
if approaching_target
    v_cmd = k1 * min(position_error, 1.0);
end
```

This mode addresses the problem of "dead reckoning" when the reference is stationary but the vehicle has not yet reached the target position. The min(position_error, 1.0) limitation prevents excessive speeds when the position error is large.

`(b.2) Parking Completed:`

```
parking_complete = (position_error < 0.3) && (v_ref < 0.01);
if parking_complete
    v_cmd = 0;
    phi_cmd = -k2 * 0.3 * etheta;
end
```

In this mode, the controller focuses exclusively on fine orientation correction, with the longitudinal velocity forced to zero. The factor 0.3 represents an attenuation of heading gain, reflecting the need for smooth movements in the final phase.

4.2 Detailed Analysis of Control Strategies

The longitudinal speed control strategy adopted can be described as feedforward-feedback control with adaptive modulation. The feedforward component:

```
v_base = v_ref + k1 * e_long;
```

represents the direct compensation term that guarantees reference velocity tracking. However, the addition of the $k_1 * e_{long}$ term transforms this component into a feedforward-feedback hybrid, where the longitudinal error modulates the reference velocity. The corrective factors:

```
lateral_factor = 1.0 / (1.0 + abs(e_lat));
heading_factor = 1.0 / (1.0 + 2.0 * abs(etheta));
v_cmd = v_base * lateral_factor * heading_factor;
```

implement a precautionary control philosophy: the vehicle automatically slows down when lateral or heading errors increase. This strategy offers significant advantages in terms of stability and safety. The steering control is the heart of the controller and integrates two separate components:

(a) Feedforward component:

```
if abs(v_cmd) > 0.1
    phi_feedforward = atan(L * thetad_dot / v_cmd);
else
    phi_feedforward = 0;
end
```

This component calculates the steering angle required to follow the desired curvature. Protection against divisions by zero when v_{cmd} is small is a design choice that prevents numerical singularities.

(b) Stanley Controller component:

```
k_stanley = k3 * (1 + abs(thetad_dot));
phi_lateral = atan2(k_stanley * e_lat, max(0.5, abs(v_cmd)));
```

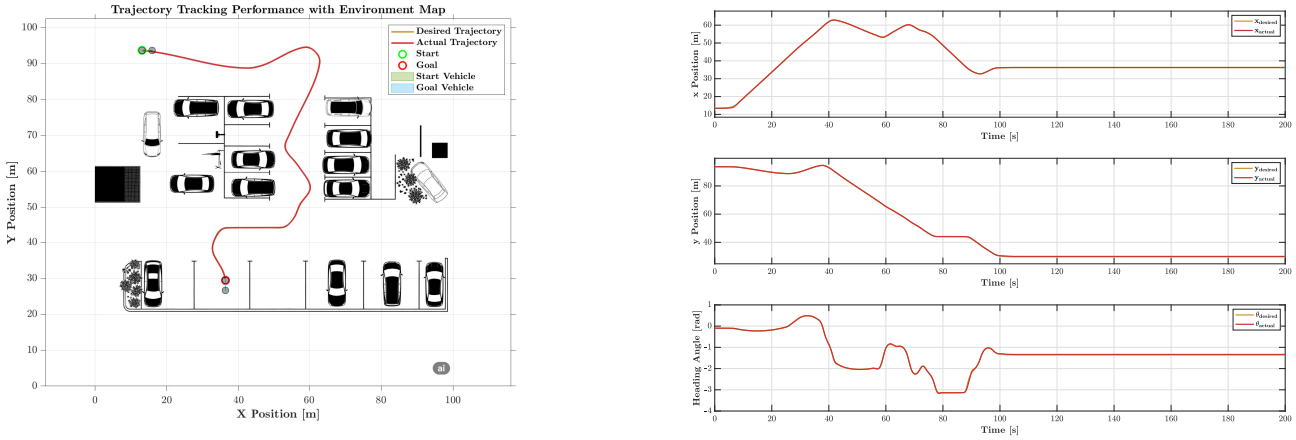
The Stanley Controller is known for its robustness in correcting lateral errors, and its mathematical formulation ensures that the steering angle is inversely proportional to speed, avoiding over-correction at high speed. The introduction of adaptive gain $k3 * (1 + abs(\dot{\theta}_d))$ is an extension of the classic Stanley Controller. The gain increases at high path curvature, improving controller responsiveness in tight turns.

4.3 Hybrid Multi-Mode Controller Performance Analysis

This section presents a comprehensive performance analysis by examining experimental results, evaluating the effectiveness of the system in terms of tracking accuracy, control stability and target convergence capability. (**Note:** The results discussed in the following refer to the implementation stored in the `FINAL_PROJECT_FSR/Hybrid_Multi_Mode_Controller` folder).

4.3.1 Trajectory Tracking Analysis

The trajectory plot (Figs.[11a]-[11b]) shows a complex path including sharp bends, significant changes of direction and a final parallel parking phase. The trajectory starts at the green point, $(x_i, y_i) = (13.15, 93.65)m$, and ends at the red point, $(x_f, y_f) = (36.35, 29.45)m$, passing through an environment with numerous obstacles represented by parked vehicles. The bicycle follows the desired trajectory (yellow line) with good accuracy along most of the route. Wide curves are handled effectively with minimal errors and the final parking phase shows good alignment with the target parking space. The controller demonstrates good navigation ability in a complex environment with static obstacles, maintaining a safe trajectory and respecting the spatial constraints imposed by the presence of parked vehicles (as can be seen by watching the video "BICYCLE_PARKING_ANIMATION_HYBRID_CONTROLLER.mp4" in the `FINAL_PROJECT_FSR/Hybrid_Multi_Mode_Controller` folder).



(a) Trajectory Tracking

(b) State Trajectory Tracking

Figure 11: Position and Orientation Tracking

4.3.2 Position and Orientation Errors

From the analysis of the graphs in Fig.[12], it follows that:

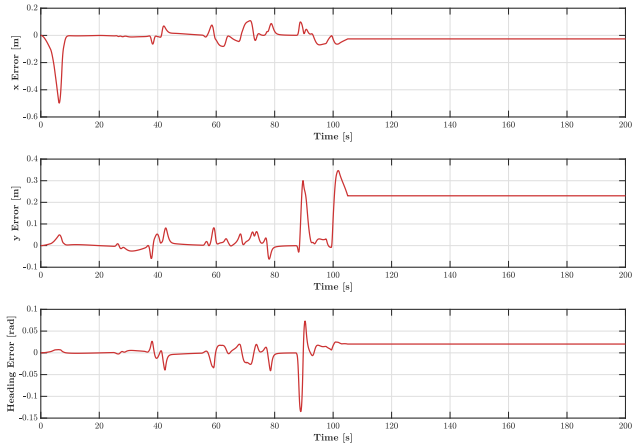


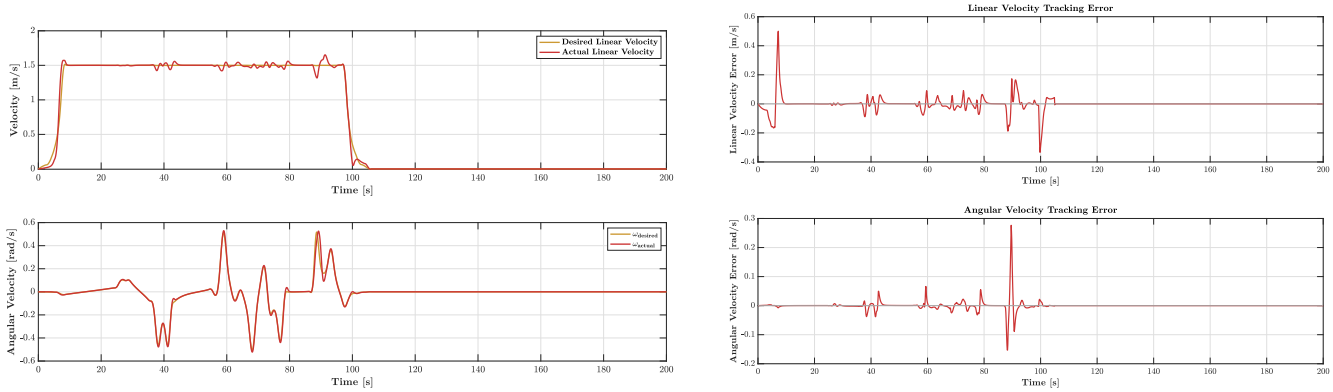
Figure 12: Position and Orientation Errors

- Longitudinal Position Error (e_x)**:** the error along the x-axis starts at approximately $-0.5m$ and quickly converges to zero, indicating effective initialization of the control system. During the cruising phase $(20 - 80s)$ the error remains within ± 0.1 m , demonstrating stable tracking during normal navigation. In the time interval $(80 - 100s)$ an error peak of up to $+0.15m$ is observed, coinciding with the most complex parking manoeuvres. In the final phase $(100 - 200s)$ the error stabilises around zero, confirming the success of the parking manoeuvre.
- Lateral Position Error (e_y)**:** The lateral error has more complex dynamics. It oscillates between $-0.1m$ and $+0.35m$, with most of the time kept below $0.1m$. A significant peak is recorded in the time interval $(80 - 100s)$: a maximum error

of around $0.35m$ coincides with the parking phase, indicating the inherent difficulty of low-speed manoeuvres in tight spaces. The error gradually decreases and stabilises around $0.23m$, a systematic offset at the final parking position.
- Orientation Error (e_θ)**:** the error remains within ± 0.003 rad for most of the time. A critical peak at time instant $90s$ is observed: a maximum error of approximately 0.08 rad during the final parking phase. The final orientation stabilises with minimal residual error, demonstrating the effectiveness of the orientation control.

4.3.3 Velocity Tracking Analysis

Consider Figs.[13a]-[13b]:



(a) Velocities Tracking

(b) Velocity Errors

Figure 13: Velocity Analysis

Looking at the first plot in Fig.[13a], the control strategy implemented clearly emerges: there is an initial acceleration phase ($0-15s$) in which the vehicle starts from a standstill and accelerates progressively until it reaches the cruising speed of $1.5m/s$. This acceleration is not instantaneous but follows a controlled profile reflecting the effectiveness of the acceleration rate limitation implemented in the code ($max\_accel = 1.5m/s$). The current speed (red line) closely follows the desired speed (yellow line) with minimal delay, demonstrating the effectiveness of feedforward control combined with feedback. This is followed by a cruise phase ($15-85s$), during which the speed remains stable around $1.5m/s$ with small variations reflecting the corrections necessary to follow the curved trajectory. It is important to note that the controller automatically adapts the speed according to the curvature of the trajectory: in tighter curves, the speed is slightly reduced to maintain stability and tracking accuracy. Finally, there is the deceleration and parking phase ($85-110s$) in which the transition from cruising speed to full stop is managed in a progressive and controlled manner. Once parking is complete, the vehicle maintains zero speed. Angular velocity (second plot in Fig.[13a]) shows more complex and dynamic behaviour than linear velocity. The angular velocity varies in a range of ± 0.6 rad/s, which is appropriate for parking manoeuvres that require significant changes of direction without being overly aggressive. The angular velocity peaks clearly correspond to the maximum curvature points visible in the spatial trajectory. This shows that the controller responds correctly to the curvature requirements of the reference trajectory. The linear velocity error (first plot in Fig.[13b]) shows distinctive characteristics that reveal the behaviour of the control system. The error reaches approximately $0.5m/s$ at the beginning, which is physiological considering that the vehicle starts from a standstill while the reference requires immediate acceleration. During cruising, the error remains mostly contained within ± 0.1 m/s, with occasional peaks of up to ± 0.2 m/s during the most demanding manoeuvres. This level of error is excellent for an autonomous vehicle control system and demonstrates the effectiveness of the feedforward v_{ref} term combined with longitudinal error correction. During the parking phase, more pronounced oscillations in velocity error are observed, with peaks reaching ± 0.3 m/s. The angular velocity error (second plot in Fig.[13b]) reveals important aspects of control. During normal navigation, the error typically remains within ± 0.1 rad/s, demonstrating precise orientation control. The error peak reaching $0.27rad/s$ represents the most challenging moment for the control system, coinciding with the manoeuvre to enter the car park where rapid and precise direction changes are required. After each error peak, the system demonstrates a rapid convergence capability, returning to minimum errors withins.

4.3.4 Control Command Analysis

Analysis of the commands generated by the controller (Fig.[14]) reveals the quality of the control strategies implemented. The speed command shows an extremely smooth profile that faithfully reflects the implemented multi-modal control strategy. During the initial phase ($0-15s$), acceleration is gradual and controlled, always remaining within the maximum limit of $2.0m/s$ without ever saturating, demonstrating the effectiveness of the acceleration rate limitation ($max_{accel} =$

$1.5m/s^2$). The transition to cruising speed occurs without significant overshooting, indicating a good balance between responsiveness and stability of the control system. The steering control effectively exploits the entire available range $\pm\pi/4$ rad/s, frequently reaching saturation limits during hard cornering, particularly visible around $60s$ and $90s$. This saturation is not problematic but indicates the optimal utilisation of the vehicle's steering capabilities to follow high curve trajectories. The controller manages these saturations without instability, keeping the trajectory tracking within acceptable limits.

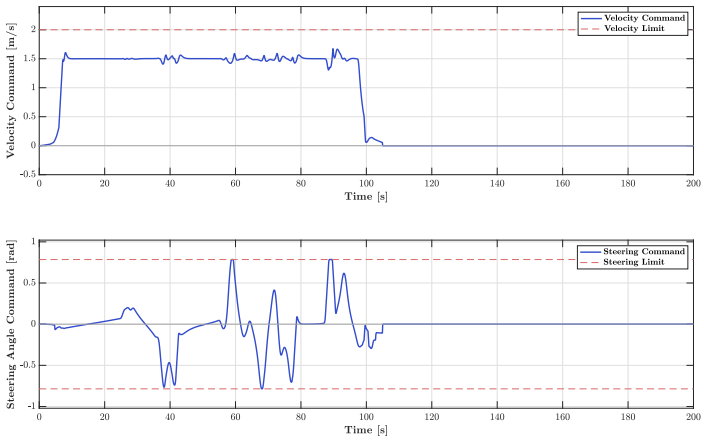


Figure 14: Control Command

5 Stanley Controller

This section describes the design and implementation of a Stanley control system for autonomous bicycle parking management. Different from the Pure Pursuit method using the rear axle as its reference point, Stanley method use the front axle as its reference point. Meanwhile, it looks at both the heading error and cross-track error. In this method, the cross-track error is defined as the distance between the closest point on the path with the front axle of the vehicle [3].

5.1 System Architecture

The control architecture consists of: a longitudinal Stanley controller, which handles movement along the direction of vehicle travel, and a lateral Stanley controller, which handles movement perpendicular to the direction of travel. Although the two controllers operate in parallel, their integration requires careful consideration of the interactions between longitudinal and lateral control. In our system, this integration occurs through various coordination mechanisms that ensure consistent behaviour of the entire control system. The direction signal serves as a communication bridge between the two controllers, allowing the lateral controller to adapt its parameters according to the direction of movement programmed by the longitudinal controller. This coordination is essential because the lateral dynamics of a bicycle change significantly between forward and reverse motion, requiring different control strategies. In addition, both controllers receive information about the current speed of the vehicle, creating a natural coupling that ensures that lateral corrections are appropriate for the speed of movement. This coupling reflects the physical reality that the manoeuvrability of a bicycle is highly dependent on its forward speed.

5.1.1 Stanley Longitudinal Controller

The longitudinal controller represents the core of the vehicle's speed control system. Its implementation is based on a Proportional-Integral (PI) controller with integrated anti-windup functionality. The controller computes the acceleration and deceleration commands required to regulate the vehicle speed according to the reference velocity and the current driving direction. The system is designed to effectively manage both forward and reverse motion. The implementation of the anti-windup mechanism prevents excessive accumulation of the integral error when actuators reach saturation, ensuring stable and predictable system performance.

5.1.2 Stanley Lateral Controller

The Stanley lateral controller implements the path following algorithm specifically adapted to the dynamic model of a bicycle. This controller calculates the steering angle required to keep the vehicle on the desired trajectory.

5.1.3 Control Strategy

The algorithm combines two intuitive actions: first, orient the vehicle towards the target point to correct the orientation error, then add an additional correction proportional to the lateral distance from the target point to ensure stable convergence. The Stanley control law implemented in the system can be expressed mathematically as:

$$\delta = \psi + \arctan\left(\frac{k \cdot e_{\text{lateral}}}{v + v_{\text{offset}}}\right)$$

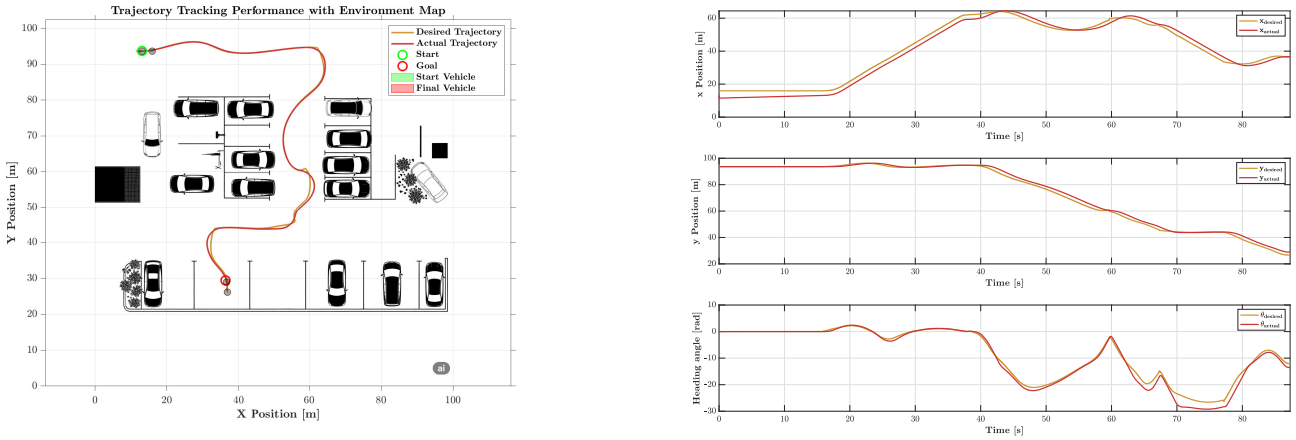
Where δ represents the commanded steering angle, ψ is the orientation error relative to the desired trajectory, k is the lateral control gain, $e_{lateral}$ is the transverse position error, v is the current vehicle speed, and v_{offset} is a speed offset to ensure numerical stability. When the transverse error is small, the control behaves almost linearly, ensuring accuracy. When the transverse error becomes significant, the arctangent function limits the maximum correction, preventing oscillations and maintaining system stability. A distinctive feature of the implementation is the intelligent two-way motion management. The system uses differentiated control gains for forward (2.0) and reverse (2.5) motion, recognising that vehicle dynamics change significantly during reverse, requiring more aggressive control to maintain stability.

5.2 Stanley Performance Analysis

A detailed analysis on the performance of the Stanley controller based on the simulation results obtained is provided below. (**Note:** The results discussed in the following refer to the implementation stored in the `FINAL_PROJECT_FSR/STANLEY` folder).

5.2.1 Trajectory Tracking Analysis

Trajectory tracking analysis reveals an overall satisfactory performance of the Stanley controller. As shown in Figs.[15a]-[15b], the vehicle follows the desired path with good fidelity, starting from the initial position and navigating through a complex environment characterised by numerous obstacles. The actual trajectory demonstrates that the controller is able to handle these complex manoeuvres while maintaining a good adherence to the target path. A particularly positive aspect is the controller's ability to navigate in tight spaces. The vehicle successfully traverses narrow corridors between parked vehicles, demonstrating control accuracy adequate for autonomous parking applications (as can be seen by watching the video "BICYCLE_PARKING_ANIMATION_STANLEY_CONTROLLER.mp4" in the `FINAL_PROJECT_FSR/STANLEY` folder).



(a) Trajectory Tracking

(b) State Trajectory Tracking

Figure 15: Position and Orientation Tracking

5.2.2 Position and Orientation Errors

From the analysis of the graphs in Fig.[16], it follows that:

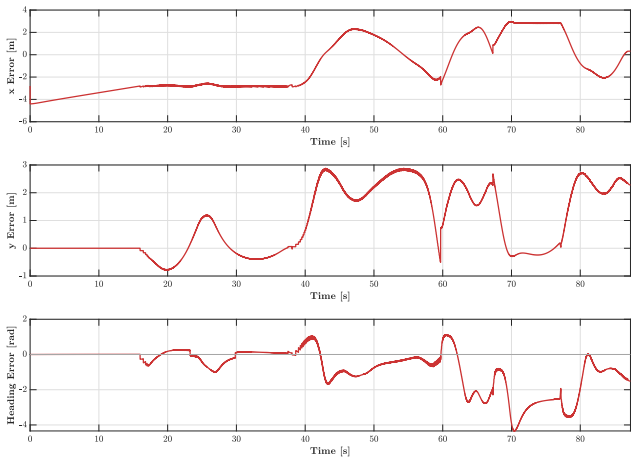
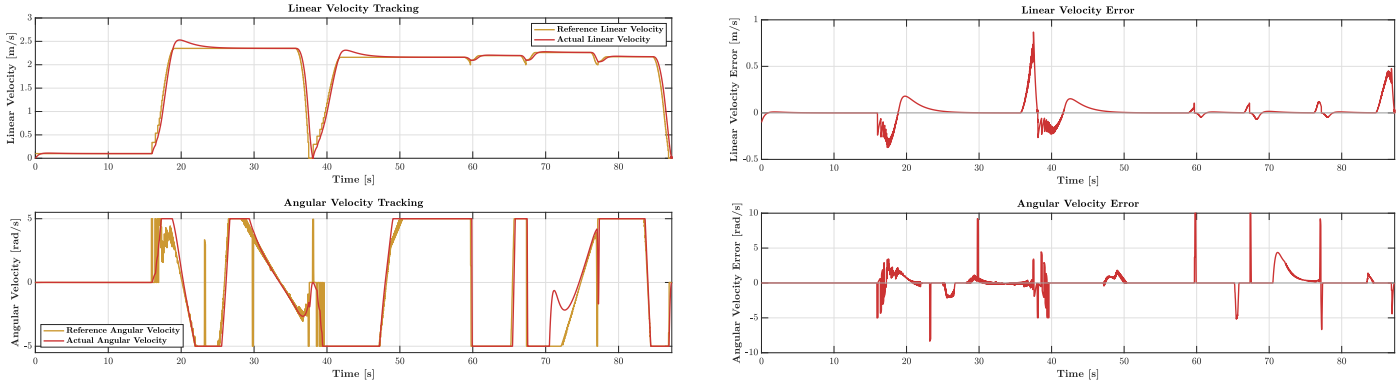


Figure 16: Position and Orientation Errors

With regard to the lateral error (or cross error), the system generally maintains values within $\pm 3\,\text{m}$. Significant peaks are observed at the tightest curves, particularly around $20s$ and between $40 - 70s$. These peaks result from the nature of the Stanley controller, which requires a certain amount of time to correct position during the most demanding manoeuvres. The longitudinal error shows interesting behaviour with variations between -1 and +3m. The general trend towards positive values suggests that the vehicle tends to lag slightly behind the desired position along the route. The orientation error shows the most significant variations, with peaks reaching 1rad and -4rad. These peaks coincide with tight path curves and represent the main challenge for the controller. The oscillatory nature of the error during curves is normal for a Stanley controller because this type of controller operates purely reactively, relying solely on instantaneous deviation from the trajectory without any anticipation capability.

5.2.3 Velocity Tracking Analysis

Consider Figs.[17a]-[17b]:



(a) Velocities Tracking

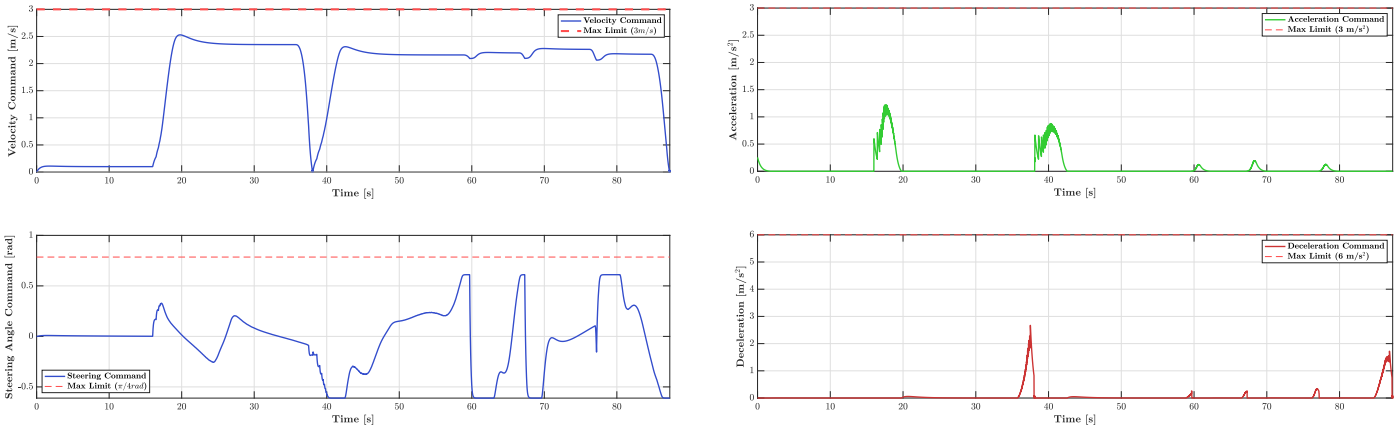
(b) Velocity Errors

Figure 17: Velocity Analysis

Linear speed tracking is virtually perfect, with the actual speed faithfully following the reference through all phases of the course. The vehicle accelerates smoothly to around $2.5m/s$, maintains this speed during straight stretches, decelerates to almost zero during the tight $40s$ turn and then regains speed for the final stretch. This precision in linear speed control is essential to ensure comfort and safety during autonomous parking. Angular speed control, on the other hand, presents more difficulties. While the system manages to follow the general reference, there are significant oscillations and spikes, particularly noticeable during tight turns. The high-frequency oscillations visible in the graph suggest that the steering control could benefit from additional filtering. Fig.[17b] confirms these observations by showing the velocity errors. The linear velocity error remains within $\pm 0.5\,\text{m/s}$ most of the time, with isolated peaks reaching 0.8 m/s only during the sharpest transitions. In contrast, the angular velocity error shows much wider variations, with peaks exceeding $\pm 10\,\text{rad/s}$.

5.2.4 Control Command Analysis

Consider Fig.[18].



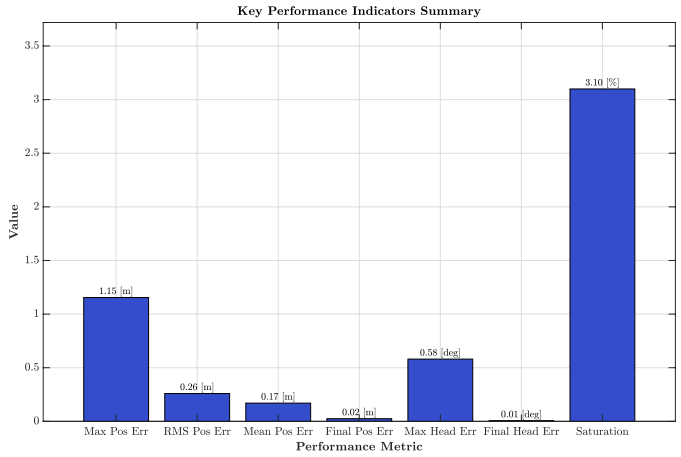
(a) (b)

Figure 18: Control Command

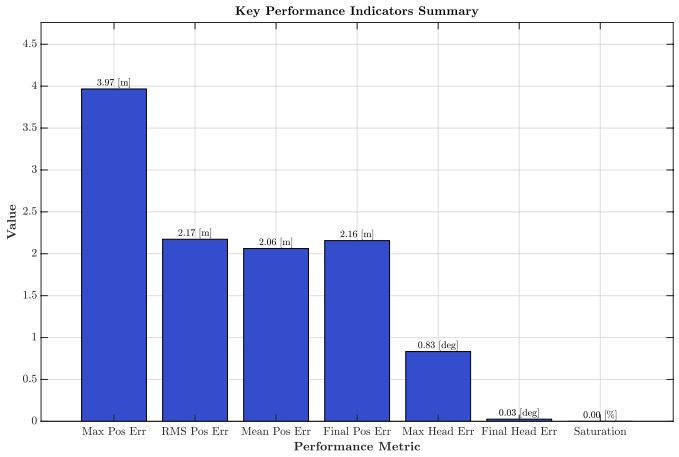
The speed controls show a well-structured profile that respects the maximum limit of $3m/s$. The controller generates smooth transitions between speed phases, avoiding abrupt changes that could compromise vehicle stability. The ability to maintain constant speeds during straight stretches and appropriately reduce speed during turns demonstrates good speed planning logic. The steering controls reveal the most critical behaviour of the system. The commands frequently oscillate between positive and negative values, with amplitudes as high as ± 0.75 rad. These oscillations, particularly evident during curved sections of the course, indicate that the controller is constantly correcting the direction. Analysis of the acceleration and deceleration commands shows generally appropriate behaviour. Accelerations are limited to around $1.2m/s^2$, well below the $3m/s^2$ limit, ensuring smooth transitions. Decelerations reach peaks of about $2.5m/s^2$ during the most intense braking, while remaining within the safety limit of $6m/s^2$. The time distribution of these commands shows that the system applies acceleration and deceleration only when necessary, minimising energy consumption and maximising comfort.

6 Comparative Performance Analysis of Autonomous Parking Controllers

The performance evaluation of four distinct controllers for autonomous bicycle parking reveals significant differences in their tracking accuracy, stability, and overall effectiveness. This analysis examines the Hybrid Controller, Model Predictive Control (MPC), Pure Pursuit, and Stanley controllers through seven key performance indicators, each providing unique insights into the controllers' behavior during parking maneuvers. The seven performance metrics evaluated include position tracking errors (maximum, RMS, mean, and final), heading errors (maximum and final), and control saturation percentage. These metrics collectively provide a comprehensive picture of each controller's ability to guide the bicycle accurately while maintaining smooth, feasible control actions.

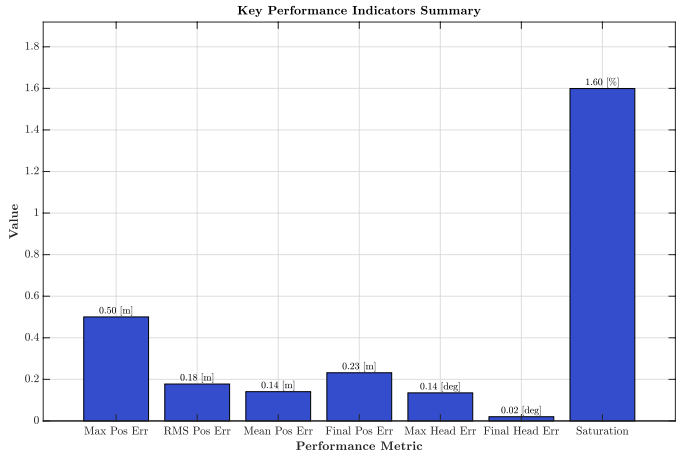


(a) Pure Pursuit

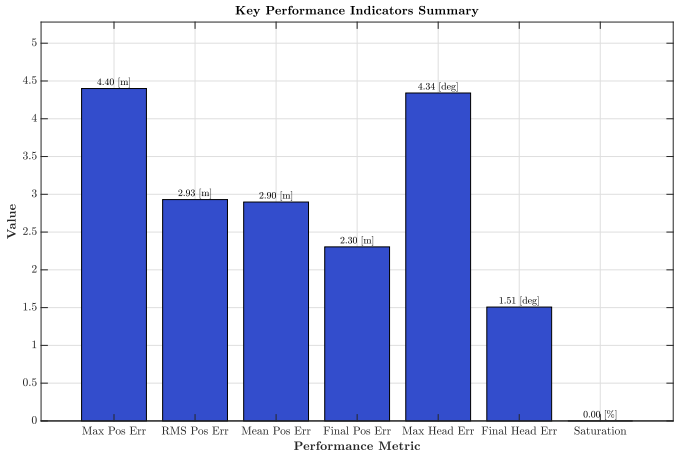


(b) MPC

Figure 19



(a) Hybrid Controlller



(b) Stanley

Figure 20

6.1 Pure Pursuit Controller Evaluation

The Pure Pursuit controller demonstrates moderate overall performance with some notable strengths. Its maximum position error of 1.15 meters places it in the middle range among the tested controllers, while the RMS error of 0.26 meters indicates relatively good average tracking performance. The mean position error of 0.17 meters suggests relatively consistent tracking performance. Where Pure Pursuit truly excels is in achieving minimal final errors. The final position error of 0.02 meters and final heading error of 0.01 degrees indicate excellent convergence to the desired parking configuration. This behavior aligns with Pure Pursuit's geometric approach, which becomes increasingly accurate as the vehicle approaches the target point. The control saturation of 3.10% is the highest among all controllers, indicating that Pure Pursuit frequently demands high control efforts. This aggressive control behavior stems from the controller's reactive nature – it attempts to minimize the lateral error to the look-ahead point without considering actuator limitations. While this contributes to its ability to achieve precise final positioning, it may result in less smooth motion profiles and increased actuator wear in practical applications.

6.2 Model Predictive Control Performance

The MPC controller exhibits a mixed performance profile. While it achieves the lowest control saturation at 0.00%, indicating that it never reaches actuator limits and maintains smooth control throughout the maneuver, its tracking performance shows notable weaknesses. The maximum position error of 3.97 meters is the highest among all controllers, suggesting that the MPC allows significant deviations from the reference path during certain phases of the parking maneuver. This behavior is characteristic of MPC's multi-objective optimization, which balances tracking accuracy

against control effort minimization. The controller appears to trade tracking accuracy for control smoothness, resulting in larger transient errors. The RMS and mean position errors (2.17 and 2.06 meters respectively) indicate persistent tracking deviations throughout the maneuver. The final position error of 2.16 meters is particularly concerning, as it suggests the controller fails to converge to the desired parking position accurately. The heading control shows better relative performance with a maximum error of 0.83 degrees and a final error of 0.03 degrees, indicating that orientation control is more effectively managed than position tracking.

6.3 Hybrid Controller Analysis

The Hybrid Controller demonstrates exceptional performance across most metrics, achieving the lowest tracking errors among all tested controllers. With a maximum position error of only 0.50 meters and an RMS position error of 0.18 meters, this controller excels at maintaining close adherence to the reference trajectory. The mean position error of 0.14 meters further confirms its consistent tracking performance throughout the parking maneuver. The heading control performance is particularly noteworthy, with a maximum heading error of 0.14 degrees and a remarkably small final heading error of 0.02 degrees. This indicates that the Hybrid Controller not only tracks the position accurately but also ensures proper orientation alignment, which is crucial for successful parking operations. However, the control saturation metric reveals a potential limitation. At 1.60%, the Hybrid Controller shows moderate saturation levels, suggesting that it occasionally demands control inputs at or near the actuator limits. This behavior likely stems from its multi-mode architecture, where transitions between different control modes (feedforward-feedback, Stanley-type lateral control and heading control) may create brief periods of aggressive control action. The supervisory logic manages these transitions, but the saturation indicates room for improvement in smoothing mode switches.

6.4 Stanley Controller Assessment

The Stanley controller demonstrates competitive performance in several key areas. With a maximum position error of 4.40m, it still allows the largest deviations from the reference trajectory among all controllers. However, the RMS and mean errors (2.93m and 2.90m respectively) are now comparable to the MPC controller, indicating similar average tracking performance. Most remarkably, the Stanley controller achieves perfect control feasibility with 0.00% saturation, matching the MPC's smoothness in control action. This is a significant finding that contradicts typical expectations for geometric controllers. The zero saturation suggests that the Stanley controller's gains have been carefully tuned to respect actuator limits while maintaining reasonable tracking performance. The heading control performance shows a maximum error of 4.34 degrees – the highest among all controllers – but recovers to a final error of 1.51 degrees. While this final heading error is larger than other controllers, it may still be within acceptable bounds for parking completion. This performance profile suggests that the Stanley controller operates with a fundamentally different strategy: it allows larger transient errors but maintains smooth, feasible control throughout the maneuver. The front-wheel position feedback mechanism characteristic of Stanley control appears to provide natural damping that prevents excessive control demands.

6.5 Comparative Analysis and Insights

The performance comparison reveals distinct controller philosophies and their practical implications:

- **Tracking Accuracy Hierarchy:** The Hybrid Controller clearly dominates with the best overall tracking metrics, followed by Pure Pursuit, then Stanley and MPC showing similar but weaker tracking performance. This hierarchy reflects the sophistication of each approach, with the multi-mode Hybrid system adapting to different phases of the parking maneuver.
- **Control Effort Philosophy:** Two distinct approaches emerge:

  - MPC and Stanley prioritize control smoothness (0% saturation), accepting larger tracking errors;

  - Hybrid and Pure Pursuit accept moderate saturation (1.60% and 3.10%) to achieve better tracking.

  This dichotomy represents a fundamental trade-off in control system design between accuracy and actuator preservation.

References

[1] Jefferson Aggor. Exploring path planning with rrt* and visualization in python. https://medium.com/@aggorjefferson/exploring-path-planning-with-rrt-and-visualization-in-python-cf5bd80a6cd6, 2023.

[2] MathWorks. Bicycle kinematic model - robotics toolbox. https://it.mathworks.com/help/robotics/ref/bicyclekinematicmodel.html, 2025.

[3] Yan Ding. Three methods of vehicle lateral control: Pure pursuit, stanley and mpc. Medium article, March 2020.

[4] Gabriel M. Hoffmann and Claire J. Tomlin. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In *Proceedings of the American Control Conference (ACC)*, pages 2296–2301, 2007.