Model Predictive Control for Frenet-Cartesian Trajectory Tracking of a Tricycle Kinematic Automated Guided Vehicle

Akash Subash [1,2], Daniel Kloeser[2], Jonathan Frey[1,3], Rudolf Reiter[1], Moritz Diehl[1,3], Karsten Bohlmann[2]

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Implemented by: Siddhesh Shingate
Northeastern University, Boston
April, 2025

***Abstract*—This project presents the implementation of a Frenet–Cartesian Nonlinear Model Predictive Control (NMPC) framework, inspired by the methodology proposed in "Model Predictive Control for Frenet-Cartesian Trajectory Tracking of a Tricycle Kinematic Automated Guided Vehicle." The approach utilizes a kinematic model and focuses on accurate trajectory tracking and obstacle avoidance in structured environments. A smooth reference path is provided as a cubic spline, and vehicle motion is represented in both Cartesian and Frenet frames to facilitate progress tracking and lateral error minimization. Constraints on control inputs, curvature, lateral deviation, and collision avoidance are incorporated directly into the predictive control formulation.**

I. Introduction

Path tracking in autonomous systems typically relies on perception-driven inputs such as lane boundaries, obstacle locations, and dynamic waypoints. In structured environments with predefined trajectories—such as factory or warehouse settings—a fixed reference path can be leveraged for improved planning and tracking. The work in [1] proposes a Frenet–Cartesian Nonlinear Model Predictive Control (NMPC) formulation that combines spatial reformulation with traditional Cartesian collision constraints to achieve effective trajectory tracking and obstacle avoidance. The Frenet frame provides a compact spatial representation of motion along a reference path, enabling structured and intuitive trajectory tracking in constrained environments.

Model Predictive Control (MPC) is particularly suitable for Automated Guided Vehicles (AGVs) in such settings, as it allows for the explicit incorporation of vehicle dynamics, constraints, and obstacle avoidance into the control formulation. [2]

Inspired by this approach, the present work implements a Frenet–Cartesian NMPC framework using a unicycle kinematic model in place of the original tricycle configuration. The model operates within velocity bounds where wheel slippage can be safely ignored, ensuring the validity of kinematic assumptions. A cubic spline is used to represent the reference path, while progress along the path and lateral deviation are handled in the Frenet frame. Obstacle constraints and collision dynamics are retained in Cartesian space. The NMPC problem is formulated and solved using CasADi and IPOPT within a receding-horizon framework, and simulation results validate the controller's ability to perform reliable path tracking and obstacle avoidance in static environments. [3]

A. Abbreviations and Acronyms

- **NMPC** — Nonlinear Model Predictive Control: A control strategy that optimizes future control actions over a time horizon, while considering nonlinear system dynamics and constraints.
- **RK4** — Runge-Kutta 4th Order: A numerical method used to accurately simulate how system states evolve over time.
- **FCF** — Frenet Coordinate Frame: A moving coordinate system that follows a reference path, making it easier to measure progress and lateral deviation.
- **CCF** — Cartesian Coordinate Frame: The standard (x, y) coordinate system used to describe position in the global map.
- **AGV** — Automated Guided Vehicle: A mobile robot designed to follow predefined paths, commonly used in warehouses and factories.
- **OCP** — Optimal Control Problem: A mathematical formulation that finds the best control actions to achieve a goal while satisfying system constraints.
- **NLP** — Nonlinear Programming: A type of optimization problem where the objective or constraints are nonlinear.
- **DAE** — Differential Algebraic Equation: A system of equations that includes both differential and algebraic parts, often used in dynamic modeling.
- **ODE** — Ordinary Differential Equation: An equation that describes how system states change continuously over time.
- **IPOPT** — Interior Point OPTimizer: A solver used to efficiently find solutions to nonlinear optimization problems with constraints.

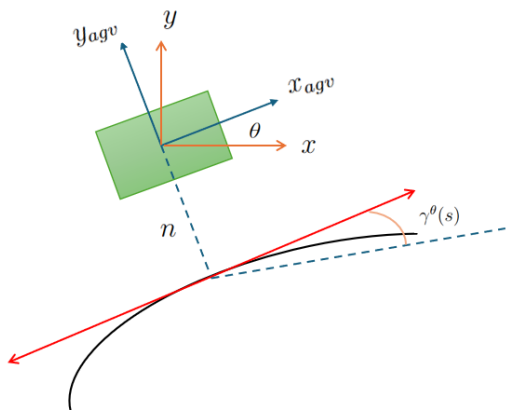

Fig. 1. Relation between Cartesian and Frenet Frame

II. System Modelling

A. Kinematics

The Cartesian state vector of the unicycle is defined as:

$$\zeta^c = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \in \mathbb{R}^3$$

representing the position and heading in the Cartesian Coordinate Frame (CCF). It is augmented by the control input vector:

$$\zeta^u = \begin{bmatrix} v \\ \omega \end{bmatrix} \in \mathbb{R}^2$$

where v is the linear velocity and ω is the angular velocity. The extended state for pose tracking becomes:

$$\zeta^p = \begin{bmatrix} x \\ y \\ \theta \\ v \\ \omega \end{bmatrix} \in \mathbb{R}^5$$

The corresponding motion model in CCF is:

$$\dot{\zeta}^p(t) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \\ a \\ \alpha \end{bmatrix}$$

where a and α are the linear and angular accelerations, respectively.

To achieve intuitive representation of path progress and deviation, the model is reformulated in the Frenet Coordinate Frame (FCF), where:

$$\zeta^f = \begin{bmatrix} s \\ n \\ \beta \end{bmatrix} \in \mathbb{R}^3$$

Here, s is the arc length along the reference path $\gamma(s)$, n is the lateral displacement from the path, and β is the heading error between the vehicle heading θ and the path tangent $\gamma^{\phi}(s)$. The path curvature is defined as $\kappa(s) = \frac{d\gamma^{\phi}(s)}{ds}$. The extended Frenet-state becomes:

$$\zeta^d = \begin{bmatrix} s \\ n \\ \beta \\ v \\ \omega \end{bmatrix} \in \mathbb{R}^5$$

In a tricycle kinematic model [**?**], steering is applied through a front wheel with a steering angle α, which introduces a decoupling between the vehicle's heading ϕ and the direction of motion. As a result, the longitudinal and lateral velocity components in the Frenet frame are scaled by $\cos(\alpha)$. This gives the dynamics:

$$\dot{s} = \frac{v\cos(\alpha)\cos(\beta)}{1 - n\kappa(s)}, \tag{1}$$

$$\dot{n} = v\cos(\alpha)\sin(\beta) \tag{2}$$

In contrast, the unicycle model directly moves in the direction of its heading θ, and turning is achieved through angular velocity ω. Since there's no steering angle, the velocity projection into the Frenet frame depends only on the heading difference $\beta = \theta - \gamma_{\phi}(s)$, giving:

$$\dot{s} = \frac{v\cos(\beta)}{1 - n\kappa(s)}, \tag{3}$$

$$\dot{n} = v\sin(\beta) \tag{4}$$

The absence of $\cos(\alpha)$ reflects the fact that the unicycle's direction of motion aligns with its body heading, unlike the tricycle, where the wheel heading and vehicle body heading can differ.

Hence the Frenet-frame dynamics:

$$\dot{\zeta}^d(t) = \begin{bmatrix} \dot{s} \\ \dot{n} \\ \dot{\beta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{v\cos(\beta)}{1-n\kappa(s)} \\ v\sin(\beta) \\ \omega - \frac{\kappa(s)v\cos(\beta)}{1-n\kappa(s)} \\ a \\ \alpha \end{bmatrix}$$

The projection from Cartesian to Frenet coordinates is defined by:

$$s^* = \underset{s}{\operatorname{argmin}} \ \|p - \gamma^p(s)\|^2$$

$$\zeta^f = F_{\gamma}(\zeta^c) = \begin{bmatrix} s^* \\ e_n^{\top}(p - \gamma^p(s^*)) \\ \theta - \gamma^{\phi}(s^*) \end{bmatrix}$$

where e_n is the unit normal vector to the path at s^*.

The inverse mapping from Frenet to Cartesian is given by:

$$\zeta^c = F_{\gamma}^{-1}(\zeta^f) = \begin{bmatrix} \gamma^x(s) - n\sin(\gamma^{\phi}(s)) \\ \gamma^y(s) + n\cos(\gamma^{\phi}(s)) \\ \gamma^{\phi}(s) + \beta \end{bmatrix}$$

The standard unicycle dynamics in CCF can be derived by differentiating the Frenet-to-Cartesian transformation:

$$\dot{\zeta}^c = \frac{dF_\gamma^{-1}(\zeta^f)}{dt} = \frac{\partial F_\gamma^{-1}(\zeta^f)}{\partial \zeta^f} f(\zeta^f, u) = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \end{bmatrix}$$

To utilize both coordinate frames for their respective advantages—Frenet for trajectory tracking and Cartesian for obstacle avoidance—we define an extended state:

$$\zeta^l = \begin{bmatrix} \zeta^c \\ \zeta^d \end{bmatrix} \in \mathbb{R}^8$$

with the combined dynamics:

$$\dot{\zeta}^l(t) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{s} \\ \dot{n} \\ \dot{\beta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \\ \frac{v\cos(\beta)}{1-n\kappa(s)} \\ v\sin(\beta) \\ \omega - \frac{\kappa(s)v\cos(\beta)}{1-n\kappa(s)} \\ a \\ \alpha \end{bmatrix}$$

B. Obstacle Avoidance Formulation

The reference paper [**?**] outlines two formulations for obstacle avoidance: one using **elliptical constraints**, and another using **multiple covering circles** to approximate the vehicle footprint. Rectangular constraints, although accurate in shape, lead to non-smooth and non-convex boundaries which reduce the feasible solution space and increase computational complexity. Elliptical constraints offer smoother, differentiable boundaries but introduce nonlinearities into the optimization problem. To simplify this, the paper suggests replacing a tight-fitting ellipse with a small number of equal-radius circles that collectively approximate the same area, thus preserving the spatial coverage while reducing the nonlinearity of the constraints.

In this project, the vehicle footprint is modeled as a **single circle** with radius $r_{\text{uni}} = 1.0$, resulting in a **quadratic constraint** for obstacle avoidance that is smooth, differentiable, and computationally efficient.

Each obstacle is modeled as a circle with known center $(x_{\text{obs}}, y_{\text{obs}})$ and radius r_{obs}. A safety buffer $\delta_{\text{safety}} = 0.05$ is included to account for modeling uncertainty.

To ensure feasibility of the optimization problem, especially under tight constraints or near obstacles, certain constraints are relaxed using slack variables. These slack variables allow the optimizer to slightly violate constraints when absolutely necessary, while penalizing such violations in the cost function to discourage them unless unavoidable.

The nominal constraint for obstacle avoidance ensures that the unicycle's collision circle (radius r_{uni}) does not intersect with any obstacle (radius r_{obs}), accounting for a safety buffer δ_{safety}:

$$\|(x_k, y_k) - (x_{\text{obs}}, y_{\text{obs}})\|_2 \geq r_{\text{uni}} + r_{\text{obs}} + \delta_{\text{safety}}$$

TABLE I
PARAMETERS

Parameter	Symbol	Value (Units)
Sampling time	dt	0.1 s
Minimum velocity	$v_{\min}$	0.1 m/s
Maximum velocity	$v_{\max}$	2.0 m/s
Minimum angular velocity	$\omega_{\min}$	$-\pi/2$ rad/s
Maximum angular velocity	$\omega_{\max}$	$\pi/2$ rad/s
Reference velocity	v_{ref}	0.8 m/s
Vehicle length	length	2.0 m
Vehicle width	width	1.0 m
Vehicle collision radius	r_{uni}	1.0 m
Lane width	lane_width	15.0 m
Maximum lateral deviation	$n_{\max}$	7.3 m
Minimum lateral deviation	$n_{\min}$	-7.3 m
Safety buffer (lateral)	–	-0.2 m
Safety buffer (obstacle)	–	-0.05 m

To soften this constraint, a non-negative slack variable $\epsilon_{\text{obs},k}$ is introduced:

$$\|(x_k, y_k) - (x_{\text{obs}}, y_{\text{obs}})\|_2 + \epsilon_{\text{obs},k} \geq r_{\text{uni}} + r_{\text{obs}} + \delta_{\text{safety}}$$

Curvature Constraint:
The geometric condition $|n_k \cdot \kappa(s_k)| < 1$ ensures the Frenet projection remains valid. To preserve feasibility under tight turns or large deviations, the constraint is softened as:

$$n_k\kappa(s_k) \leq 0.99 + \epsilon_{\text{curv},k}$$
$$n_k\kappa(s_k) \geq -0.99 - \epsilon_{\text{curv},k}$$

Lateral Deviation Constraint:
Lateral bounds ensure that the unicycle remains within lane limits. These are softened using $\epsilon_{\text{lat},k}$ as:

$$n_k \leq n_{\max} + \epsilon_{\text{lat},k}$$
$$n_k \geq n_{\min} - \epsilon_{\text{lat},k}$$

Summary of Soft Constraints:

- Curvature constraint: $|n_k \cdot \kappa(s_k)| \leq 0.99 + \epsilon_{\text{curv},k}$
- Lateral deviation: $n_{\min} - \epsilon_{\text{lat},k} \leq n_k \leq n_{\max} + \epsilon_{\text{lat},k}$
- Obstacle distance: $\|(x_k, y_k) - (x_{\text{obs}}, y_{\text{obs}})\|_2 \geq r_{\text{uni}} + r_{\text{obs}} + \delta_{\text{safety}} - \epsilon_{\text{obs},k}$

All slack variables $\epsilon_{\cdot,k}$ are constrained to be non-negative:

$$\epsilon_{\cdot,k} \geq 0$$

and are penalized in the cost function as:

$$J_{\text{slack}} = \sum_{k=0}^{N} (w_{\text{lat}} \cdot \epsilon_{\text{lat},k} + w_{\text{curv}} \cdot \epsilon_{\text{curv},k}) + \sum_{k=1}^{N} \sum_{i=1}^{n_{\text{obs}}} w_{\text{obs}} \cdot \epsilon_{\text{obs},k,i}$$

This formulation strikes a balance between robustness and flexibility, ensuring constraint satisfaction while maintaining feasibility in complex scenarios.

III. Controller Methodology

A. Lifted Frenet–Cartesian Formulation

The overall problem can be seen as a lifted Frenet Optimal Control Problem. The idea here is to use both Frenet and Cartesian coordinate frames together, each where it makes

the most sense. The Frenet frame handles what really matters for tracking the path—things like how far along the path the vehicle is (s), how far off it is laterally (n), and how misaligned its heading is compared to the path (β).

At the same time, the Cartesian frame is still used for things like obstacle avoidance and collision checking—because those tasks need global position information, and it's cleaner to handle them that way.

By lifting the problem into this combined form, we avoid constant switching between coordinate systems. That switching can introduce numerical errors and make the math messy. Instead, the controller just works with one big state vector that includes all the relevant parts from both frames—only the pieces we actually need. This reduces indexing overhead, keeps the optimization problem focused, and makes the cost terms easier to interpret and tune. It's a neat way to get the best of both worlds: the structure of the Frenet frame and the precision of the Cartesian one.

B. NMPC Formulation

The NMPC problem is formulated as:

$$\min_{u_0,\zeta_0,\ldots,u_{N-1},\zeta_N} \sum_{k=0}^{N-1} L(\zeta_k, u_k) + E(\zeta_N)$$

subject to:
current state and estimated current state

$$\zeta_0 - \bar{\zeta}_0 = 0$$

dynamics

$$\zeta_{k+1} - \phi(\zeta_k, u_k) = 0 \quad k = 0, \ldots, N-1$$

control bounds

$$v_{\min} \leq v_k \leq v_{\max}, \quad \omega_{\min} \leq \omega_k \leq \omega_{\max}$$

geometric bounds

$$n_{\min} \leq n_k \leq n_{\max}, \quad n_k \kappa(s_k) \leq 0.99$$

obstacle avoidance

$$\|(x_k, y_k) - (x_{\text{obs}}, y_{\text{obs}})\|_2 \geq r_{\text{uni}} + r_{\text{obs}} + \delta_{\text{safety}}$$

Here, ζ_k is the augmented state vector containing both Cartesian and Frenet variables, $u_k = [v_k, \omega_k]^\top$ is the control input at time step k, and $\phi(\zeta_k, u_k)$ denotes the discrete-time dynamics obtained via Runge-Kutta integration.

C. Cost Function Design

The stage cost $L(\zeta_k, u_k)$ is composed of multiple weighted objectives:

$$\begin{aligned} L(\zeta_k, u_k) = &- Q_s(s_{k+1} - s_k) + Q_{s,\text{ref}}(s_k - s_{k,\text{ref}})^2 + Q_n n_k^2 \\ &+ Q_\theta \sin^2(\theta_k - \gamma^\phi(s_k)) + R_v(v_k - v_{\text{ref}})^2 + R_\omega \omega_k^2 \end{aligned}$$

Where:

- $-Q_s(s_{k+1} - s_k)$ encourages forward progress.
- $Q_{s,\text{ref}}(s_k - s_{k,\text{ref}})^2$ penalizes deviation from the desired reference path values.
- $Q_n n_k^2$ penalizes lateral deviation from the path.
- $Q_\theta \sin^2(\theta_k - \gamma^\phi(s_k))$ penalizes heading misalignment.
- $R_v(v_k - v_{\text{ref}})^2$ encourages tracking a desired linear velocity.
- $R_\omega \omega_k^2$ penalizes high angular velocities.

To promote smooth control profiles, control rate penalties are included:

$$L_{\text{rate}}(u_k, u_{k-1}) = w_v(v_k - v_{k-1})^2 + w_\omega(\omega_k - \omega_{k-1})^2$$

The terminal cost is defined as:

$$E(\zeta_N) = Q_n n_N^2 + Q_{s,\text{ref}}(s_N - s_{N,\text{ref}})^2 + 2Q_\theta(\theta_N - \gamma^\phi(s_N))^2$$

D. Implementation

The NMPC problem is implemented using the CasADi optimization framework with the IPOPT solver. At each control step, the nonlinear problem is solved with the current state $\bar{\zeta}_0$ as input, and only the first control input u_0 is applied to the system in a receding horizon fashion.

The dynamics are discretized using a 4th-order Runge-Kutta (RK4) integration scheme:

$$\begin{aligned} k_1 &= f(\zeta_k, u_k) \\ k_2 &= f(\zeta_k + \frac{dt}{2} k_1, u_k) \\ k_3 &= f(\zeta_k + \frac{dt}{2} k_2, u_k) \\ k_4 &= f(\zeta_k + dt \cdot k_3, u_k) \\ \zeta_{k+1} &= \zeta_k + \frac{dt}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

E. Process Overview

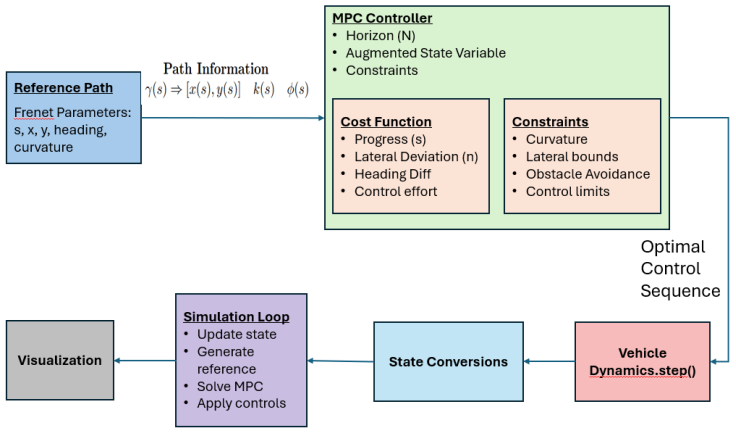

Fig. 2. Block Diagram

The control pipeline for trajectory tracking in this project follows the architecture, as shown in Fig. 2. The system begins with a *Reference Path*, defined as a smooth centerline using cubic splines. This path provides essential geometric properties such as the global coordinates

$$\gamma(s) = [x(s), y(s)],$$

heading $\phi(s)$, and curvature $\kappa(s)$. These Frenet-based quantities serve as the primary reference against which the vehicle's performance is evaluated.

This information flows into the *MPC Controller*, which operates over a finite prediction horizon N. The controller receives the current state of the vehicle and the reference path parameters, and formulates a constrained optimal control problem using an augmented state vector. This lifted state combines Frenet components—progress s, lateral deviation n, and heading error β—with Cartesian coordinates x, y, and θ, enabling the controller to reason simultaneously about spatial alignment and obstacle proximity. The cost function is structured to encourage path-following behavior through progress maximization and deviation penalties, while the constraints enforce velocity limits, curvature regularity, lane bounds, and safe obstacle clearance.

The optimal control sequence generated by the MPC is passed to the *Vehicle Dynamics* module, which simulates unicycle motion using its kinematic equations. This simulation step is executed using a fourth-order Runge-Kutta (RK4) integration scheme to maintain numerical accuracy. The resulting state is passed through the *State Conversion* module, which transforms the updated Cartesian state back into the Frenet frame. This enables precise tracking of the vehicle's progress along the path and its lateral behavior.

Within the *Simulation Loop*, the state is updated, a new reference segment is generated, the MPC problem is resolved, and the first control input is applied—realizing a classic receding horizon strategy. The loop continues until the vehicle completes the trajectory or the simulation terminates. Finally, the results—trajectory, obstacle interactions, and heading behavior—are relayed to the *Visualization* block for interpretation and analysis.

Importantly, this entire control architecture is an instance of a *Lifted Frenet Optimal Control Problem*, where trajectory tracking is formulated in the Frenet frame for interpretability and structure, while collision avoidance is handled in the Cartesian frame for geometric accuracy. This lifting improves numerical tractability by reducing unnecessary coordinate transformations and allows each subsystem—tracking and avoidance—to operate within its most natural representation. By decoupling along-track and cross-track behavior, the controller gains both precision and robustness in navigating structured environments with static obstacles.

IV. Results

The effectiveness of the implemented Frenet–Cartesian NMPC controller is validated through simulation results over a structured reference path with multiple circular obstacles. The controller demonstrates accurate trajectory tracking, smooth control actions, and successful avoidance of obstacles.

A. Path Tracking and Obstacle Avoidance

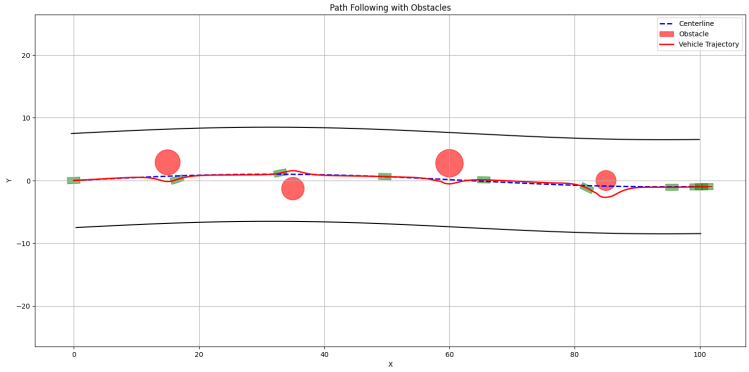

Fig. 3. Trajectory with 4 obstacles

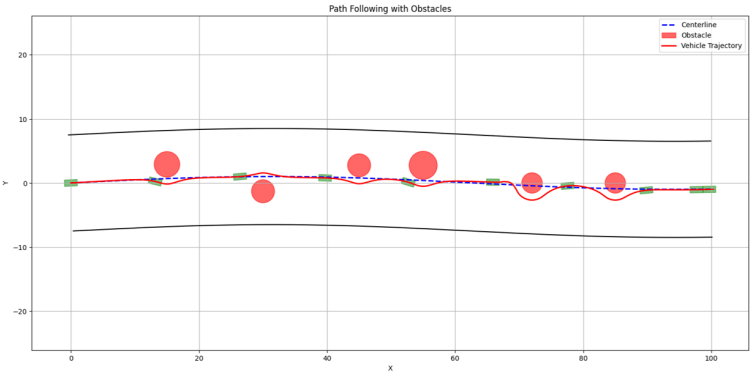

Fig. 4. Trajectory with more frequent obstacles

Fig. 3 and Fig. 4 show the successful execution of trajectory tracking, where the vehicle follows the reference path (blue dashed line) while avoiding all obstacles (red circles). To evaluate the controller's robustness, obstacles were deliberately placed along the path to introduce challenging navigation scenarios, rather than simplifying the task for the unicycle model. As seen in Fig. 4, the vehicle performs smooth avoidance maneuvers without deviating excessively from the reference, validating the effectiveness of the controller in resolving the nonlinear Optimal Control Problem (OCP) under both geometric and dynamic constraints.

B. State and Control Progressions

Fig. 5 illustrates the evolution of key state variables and control inputs throughout the simulation:

- **Path progress (s):** The arc length variable increases monotonically with simulation steps, showing consistent forward motion. The full 100 m path is covered in approximately 600 steps.

- **Lateral deviation (n):** The lateral offset remains tightly regulated near zero, with deviations bounded within $\pm 1.8\,\mathrm{m}$. Notable peaks in n are observed around $s = 40\,\mathrm{m}$ and $s = 80\,\mathrm{m}$, where the vehicle performs avoidance maneuvers.
- **Heading (θ):** The vehicle heading in Fig. 6 exhibits smooth yet sharp reactive changes to adapt to obstacle-induced path deviations and curvature.
- **Control Input (u):** Both linear and angular velocities remain within their respective bounds throughout the simulation (Figs. 7 and 8). The linear velocity v stays near the upper limit of $2.0\,\mathrm{m/s}$ for most of the trajectory, with temporary reductions near high-curvature segments and the terminal portion to facilitate safe turning. The angular velocity ω exhibits sharp but bounded variations—particularly around $s = 20\,\mathrm{m}$, $40\,\mathrm{m}$, $60\,\mathrm{m}$, and $80\,\mathrm{m}$—corresponding to obstacle interactions. Peak values of $|\omega| \approx 1.5\,\mathrm{rad/s}$ highlight the controller's ability to execute agile maneuvers while maintaining feasibility.

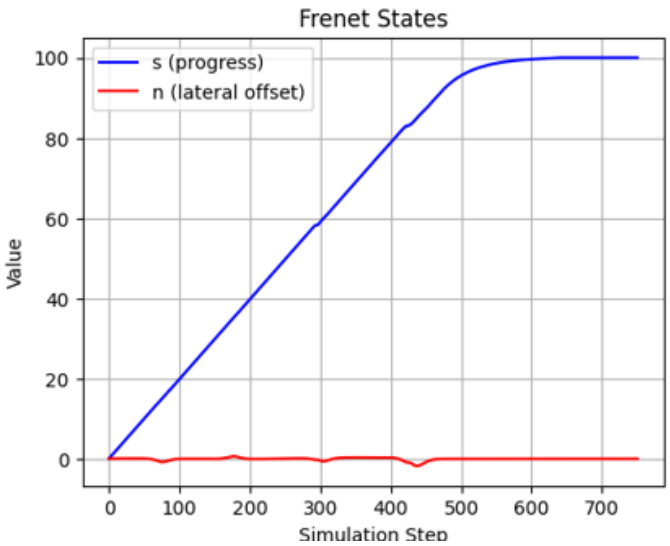

Fig. 5. Variation in lateral deviation and arc length

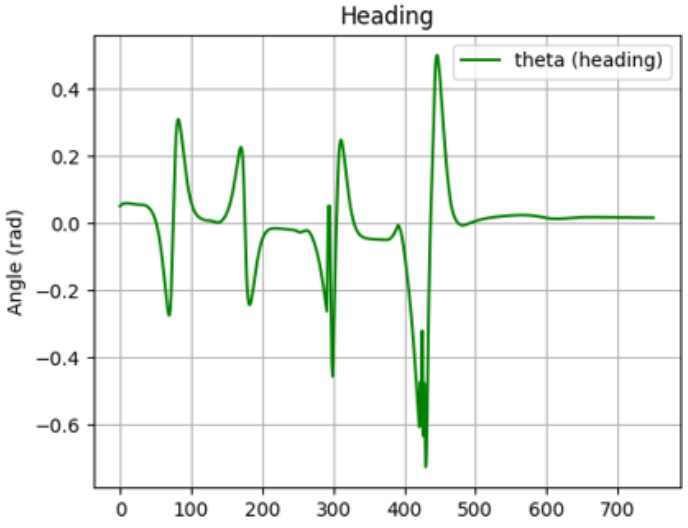

Fig. 6. Heading changes

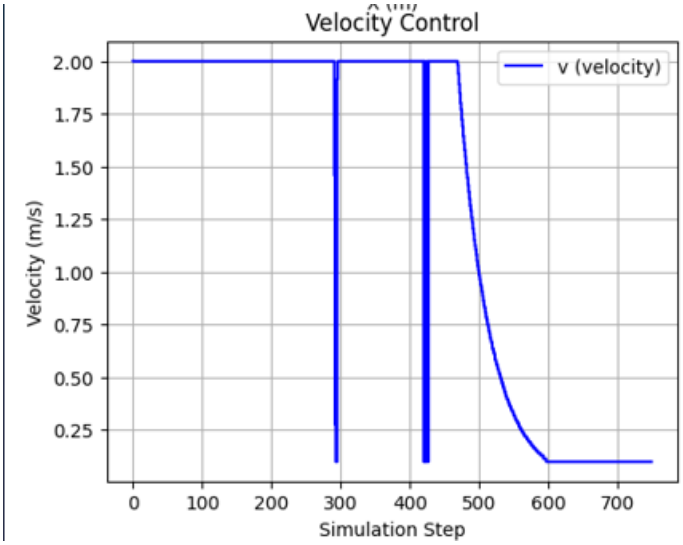

Fig. 7. Linear Velocity

C. Computational Performance

Solver performance, as depicted in Fig. 9 (bottom-left), reflects the real-time feasibility of the NMPC formulation:

- **Initial segments (low curvature, obstacle-free):** Solver times are consistently below $1000\,\mathrm{ms}$.
- **High complexity zones (obstacle-dense, curved path):** Peak computation times reach $10{,}000\,\mathrm{ms}$ due to tighter constraint enforcement and slack variable optimization.
- **Average solver time:** Approximately 4000–$5000\,\mathrm{ms}$ per iteration, which is acceptable for a $100\,\mathrm{ms}$ control interval in simulation, but may necessitate further optimization for real-time embedded deployment.

V. Conclusion

By leveraging the spatial structure of the Frenet frame and retaining obstacle constraints in Cartesian space, the controller achieves smooth path following and effective collision avoidance within a constrained environment.

The simulation results validate the controller's ability to maintain low lateral and heading errors, dynamically adjust velocity and steering in response to obstacles, and progress steadily along a predefined path. The lifted formulation combining both coordinate frames allows for intuitive control design and structured constraint enforcement.

However, the current implementation can be further improved in several ways:

- **Vehicle Dynamics:** Extending the model to incorporate higher-fidelity dynamics, such as a tricycle or bicycle model with actuation delay or tire slip, can make the framework applicable to more realistic driving scenarios.
- **Trajectory Smoothness:** Generating reference paths with continuous curvature or optimizing for curvature regular-

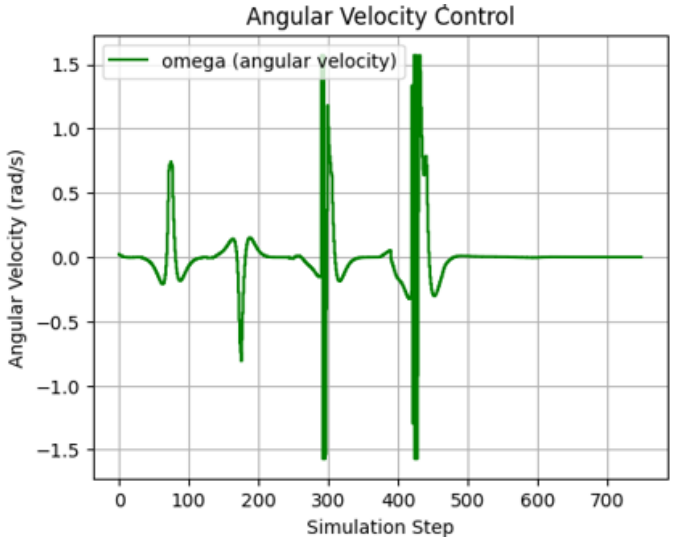

Fig. 8. Angular Velocity

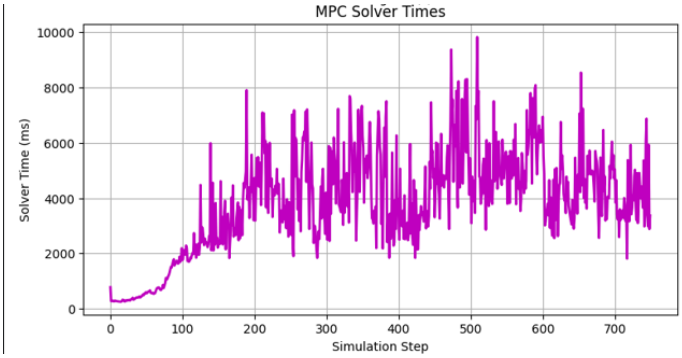

Fig. 9. Solver Time

ity can lead to smoother and more efficient maneuvers rather than sharp turns as observed in the results.

- **Real-time Performance:** To meet real-time constraints on embedded platforms, further optimization of solver time is essential—this may include reducing horizon length, using tailored solvers, or employing warm-start strategies.
- **Dynamic Environments:** Integrating perception-based dynamic obstacle handling and re-planning mechanisms could extend applicability to real-world autonomous navigation.

Overall, the framework provides a strong foundation for modular trajectory tracking with structured constraints and sets the stage for future work involving more complex dynamics, adaptive planning, and real-time control deployment.

VI. Resources

GitHub Repository:
https://github.com/Siddhesh582/Cartesian-Frenet-based-MPC.git

Simulation Video:
https://youtu.be/JKJpPWUT9_w

References

[1] A. Subash, D. Kloeser, J. Frey, R. Reiter, M. Diehl, and K. Bohlmann, "Model Predictive Control for Frenet-Cartesian Trajectory Tracking of a Tricycle Kinematic Automated Guided Vehicle," in 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2024, pp. 11948-11953, doi: 10.1109/IROS58592.2024.10802822.

[2] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét Frame," in 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 987-993, doi: 10.1109/ROBOT.2010.5509799.

[3] C.-L. Chen, J. Li, M. Li, and L. Xie, "Model Predictive Trajectory Tracking Control of Automated Guided Vehicle in Complex Environments," in 2018 IEEE 14th International Conference on Control and Automation (ICCA), 2018, pp. 405-410, doi: 10.1109/ICCA.2018.8444247.