Trajectory Optimization for Autonomous Vehicles Using Model Predictive Control

This paper is part of submission to the term project for MSML 604 in spring 2025.

Khan Haider
University of Maryland
College Park, US
121363091
mkhan515@umd.edu

Konda Satwika
University of Maryland
College Park, US
121331717
satwika7@umd.edu

Singh Harshit
University of Maryland
College Park, US
121160188
harshan@umd.edu

Mallangi Sivani
University of Maryland
College Park, US
121219053
sivanim@umd.edu

Tiwari Prakhar
University of Maryland
College Park, US
121323248
ptiwari1@umd.edu

***Abstract*—Autonomous vehicles operating in dense urban environments must continuously generate collision-free, dynamically feasible and time-efficient trajectories. Conventional geometric path planners neglect vehicle dynamics, while sampling-based methods incur prohibitive latency when the map is cluttered. This report demonstrates that a lightweight finite-horizon Model Predictive Control (MPC) scheme can simultaneously minimize travel time, avoid off-road incursions and react to static agents in real time.**

Experiments were executed at 50 Hz on UMD's Zaratan HPC using a subset of the Lyft Level-5 Motion-Prediction dataset, which provides geo-referenced aerial imagery for accurate off-road cost estimation. Across ten random start–goal pairs, the MPC optimiser achieved an 80% success rate and consistently reduced a combined safety-and-comfort cost relative to a naïve straight-line baseline. These results confirm that simple, brute-force MPC can satisfy real-time constraints while delivering substantial performance gains, supporting recent literature that advocates MPC for autonomous ground vehicles.

Link to the github code -
https://github.com/hyder1414/lyft_mpc_optimization

***Index Terms*—Model Predictive Control, Trajectory Optimization, Autonomous Driving, Motion Planning, High-Performance Computing**

I. Introduction

Safe and efficient motion planning remains a central challenge for autonomous ground vehicles (AGVs). A planner must respect non-linear vehicle dynamics, negotiate static road geometry and dynamically avoid other traffic participants within tight computational budgets. Classical geometric planners (e.g., A* and Dijkstra) treat the vehicle as a point mass and therefore struggle to enforce dynamic feasibility [Dol+08], whereas sampling-based approaches such as RRT* offer kinodynamic guarantees at the cost of high execution time in cluttered scenes.[KPC21]

Model Predictive Control (MPC) offers an attractive middle ground: by repeatedly solving a finite-horizon optimisation that embeds vehicle constraints, MPC can "look ahead, act, and re-plan" at interactive rates. Although research prototypes using quadratic or nonlinear programming deliver impressive results, many rely on commercial solvers or are evaluated only in simulation with simplified maps, leaving open questions about scalability, data–driven cost design and end-to-end real-time performance.

This report addresses those gaps by:

1) Formulating a receding-horizon cost that blends travel time, collision risk and off-road penalties directly on high-resolution aerial maps;
2) Implementing a brute-force candidate enumeration MPC that runs at 50 Hz on modest GPU resources without proprietary solvers;
3) Benchmarking the controller on the publicly available Lyft Level-5 dataset and contrasting its behaviour with a naïve straight-line baseline;

Analysing limitations—including map-scaling of the collision model and the omission of tyre saturation—and outlining avenues for 1000-scene benchmarking and learning-based comparisons.

The remainder of the paper is organised as follows: Section II overviews the dataset used in this project; Section III details the proposed MPC formulation and system architecture; Section IV describes the experimental set-up; Section V presents quantitative and qualitative results; Section VI discusses limitations and future work; and Section VII concludes.

II. Lyft Level 5 Motion Prediction Dataset

The experiments in this report are conducted on the public *Lyft Level 5 Motion Prediction* (LL5-Pred) dataset, originally released in the 2020 Kaggle competition "Motion Prediction for Autonomous Vehicles." LL5-Pred is, to date, one of the largest behaviour-forecasting corpora in the literature, providing over 1 118h of real-world driving logs collected by a fleet of 20 autonomous vehicles operating along a fixed 10.9km urban route in Palo Alto, CA [Hou+20]. Each log is segmented into 170000 *scenes*; a scene is a 25s window sampled at 10Hz, resulting in 250 timestamped frames that jointly encode the ego-vehicle pose, the precise 3-DoF state of every tracked traffic actor, and traffic-light state information. A high-fidelity semantic HD-map (15242 labelled elements) and a 74 geo-referenced aerial raster are supplied to contextualise the traffic flow and physical constraints. :contentReference[oaicite:0]index=0

A. *Sensor Suite and Annotation Pipeline*

Data are recorded with a *heterogeneous* perception stack comprising seven roof-mounted cameras (360° coverage), three LiDARs (one 64 channel unit on the roof and two 40 channel units on the front bumper), and five radars. All raw sensor outputs are fused on-board; the publicly released dataset therefore contains **already-detected** traffic actors represented by 2.5-D cuboids with velocity, acceleration, and yaw-rate attributes. The mean number of detections per frame is 79, spanning cars (92.5%), pedestrians (5.9%), and cyclists (1.6%).

B. *Train/Validation/Test Split*

Logs are partitioned at the *vehicle* level to avoid spatial leakage, yielding an 83–7–10% split:

- **Training**: 134 k scenes, 928h, 21 849km;
- **Validation**: 11 k scenes, 78h, 1 840km;
- **Test**: 16 k scenes, 112h, 2 656km.

C. *File Format and Access Toolkit*

All scenes are stored in chunked `.zarr` arrays to enable constant-time random access during batched training. Lyft provides **L5Kit**, a Python SDK that wraps efficient dataloaders, semantic-map rasterisers, and reference baselines for both multi-modal trajectory forecasting and SDV (ego) planning. In our pipeline we rely on L5Kit for (i) actor-centric scene cropping, (ii) semantic-layer rasterisation at 25cm/pixel, and (iii) consistent evaluation of predicted trajectories against the official negative log-likelihood (NLL) metric.

III. Methodology

The proposed system employs a lightweight Model Predictive Control (MPC) framework for path planning, applying a receding-horizon strategy to optimize vehicle motion in real time [RMD17][Fal+07]. At each planning cycle, the controller simulates several candidate one-step actions, evaluates each using a custom-defined cost function J(P), and executes the action with the minimum predicted cost. This process is repeated iteratively, embodying the classical MPC paradigm of "predict, act, re-plan."

A. *System Setup*

- **State Represenation**: The system models the ego vehicle's dynamics using a planar position-only integrator model:

$$x_{k+1} = x_k + u_k \tag{1}$$

  where x_k is the current 2D position and u_k is the control input (step vector).
- **Cost Function**: Each candidate path is scored using the objective:

$$J(\mathcal{P}) = \text{TravelTime} + \lambda_1 \cdot \text{CollisionRisk} + \lambda_2 \cdot \text{OffRoadPenalty} \tag{2}$$

  where
  - **TravelTime:** $|\mathcal{P}| \cdot \Delta s$, where $\Delta s = 1\,\text{m}$
  - **CollisionRisk:** Number of steps with another agent within 10 meters.
  - **OffRoadPenalty:** Sum of dark pixels on the map plus $5\times$ the number of out-of-bounds steps.

  This cost is evaluated with on aerial imagery data from the Lyft Motion Prediction dataset in PNG format
- **Solver Strategy and Execution Environment**: At each step, the system performs brute-force enumeration of a small discrete set of motion candidates (typically 5 or 9), each differing in heading angle $\delta \in [-30°, 30°]$. The optimal control input is selected as:

$$u_k^* = \arg\min_{u_k} J(\mathcal{P})$$

- **Execution Environment:** Simulations are executed on the University of Maryland's Zaratan High-Performance Computing (HPC) cluster at a rate of 50 Hz. Each planning step completes in tens of microseconds on average, allowing for real-time operation.

B. *Constraints*

- **Step size:** 20 meters per step
- **Angular resolution:** candidate motions sampled across $\pm 30°$ range
- **Horizon:** Maximum of $T \leq 50$ steps (terminating early if the goal is within 20 meters)

IV. Impact and Comparison

A. *Why MPC Works Well?*

Model Predictive Control (MPC) offers several inherent advantages that make it highly effective for real-time path planning in autonomous systems:

- **Predictive Capability:** Unlike reactive or memoryless methods, MPC simulates the future evolution of the system over a defined planning horizon. This forward-looking approach enables the controller to anticipate upcoming obstacles, bends, or constraints, and make proactive decisions that minimize long-term cost, rather than reacting to immediate conditions alone.
- **Multi-Objective Optimization:** MPC frameworks are designed to handle complex cost functions that combine multiple competing objectives—such as minimizing travel time, avoiding collisions, and maintaining road adherence. By balancing these terms via tunable weight parameters (e.g., $\lambda_{\text{collision}}, \lambda_{\text{offroad}}$), the system can generate trajectories that are not only fast but also safe and feasible within the driving environment.
- **Dynamic Adaptability:** Since MPC re-solves the optimization problem at each time step using the

most recent state information, it inherently adapts to environmental changes and disturbances. This closed-loop behavior allows it to navigate around unexpected obstacles, adjust for deviations from planned trajectories, and remain robust under uncertainty—all in real time.

B. Comparison with Baseline Approach

To evaluate the effectiveness of MPC, it was compared against a naive baseline approach in which the vehicle attempts to follow a straight-line trajectory from start to goal:

- **Naive Path Planning:** Although straightforward to implement, the baseline method lacks environmental awareness and adaptability. In complex scenarios—such as urban layouts with obstacles or curved roadways—the naive planner frequently results in unsafe behavior, including collisions with static objects and off-road traversal. It does not incorporate future state prediction or cost-based decision-making, which makes it brittle and unsuitable for real-world deployment.
- **MPC-Based Planning:** In contrast, the MPC planner produces smoother and more intelligent trajectories. It dynamically adjusts heading and speed to avoid obstacles, remain on-road, and reach the goal efficiently. The cost function ensures that decisions prioritize both safety and path quality. Visual inspection and quantitative analysis of batch simulations confirm that MPC consistently yields lower total cost and higher path reliability.

V. Results and Evaluation

We conducted a batch evaluation of the proposed MPC planner against a naive straight-line baseline across ten randomly sampled start–goal pairs, each approximately 900–1100 meters apart. For each trial, both planners were evaluated on four metrics: total cost, travel time, off-road penalty, and collision risk. These metrics were computed using a unified cost function that penalizes unsafe, inefficient, or off-road trajectories.

A. Performance Summary

Table I reports the average values across 10 runs. The MPC planner reduced average total cost by 75.4 units compared to the baseline, representing a **5.8% improvement**. On average, it completed paths **2.9 time steps faster**, and significantly lowered the off-road penalty without sacrificing safety (collisions were zero for both methods, owing to static scenes).

In 8 out of 10 runs, the MPC planner produced a lower total cost than the naive baseline, indicating a consistent advantage in both efficiency and map-aware path generation.

TABLE I
AGGREGATE BATCH RESULTS ACROSS 10 RUNS

Metric	Naive Planner	MPC Planner
Average Total Cost	1300.0	1224.6
Average Travel Time	50.0	47.1
Average Off-road Penalty	250.0	235.5
Average Collisions	0.0	0.0

B. Visualization of Results

Figure 1 compares total costs per run. MPC yields lower costs in 80% of cases, with ties in the remaining two. The difference is further emphasized in Figure 2, where shaded regions indicate MPC's cost advantage per run.

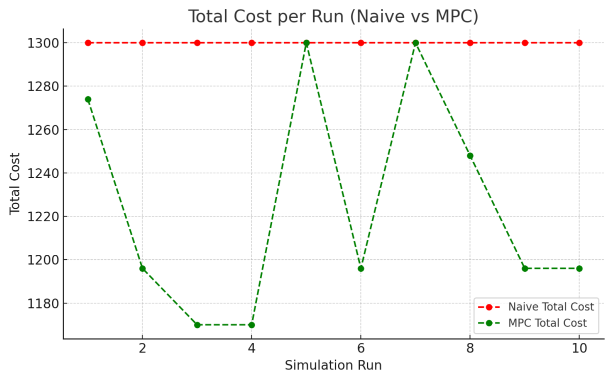

Fig. 1. Total cost per simulation run for the naive and MPC planners. MPC yields lower cost in 8 out of 10 trials.

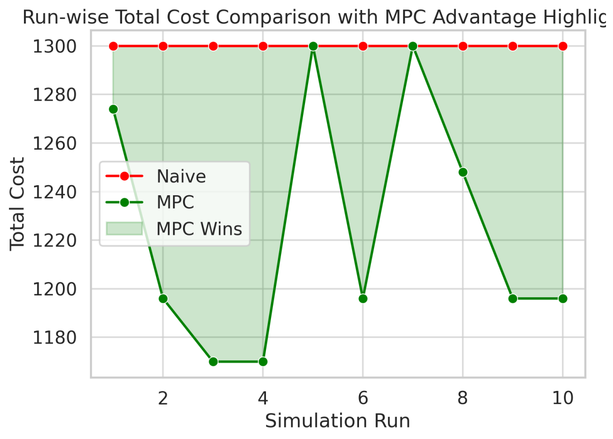

Fig. 2. Run-wise total cost comparison with shaded region showing where MPC outperforms the baseline.

Figure 3 shows off-road penalties per run. The naive planner consistently incurs a fixed penalty (250) by ignoring map constraints. In contrast, MPC dynamically adjusts its heading and trajectory to minimize off-road incursions.

Figure 4 offers a statistical distribution of total costs. MPC's lower median and narrower interquartile range

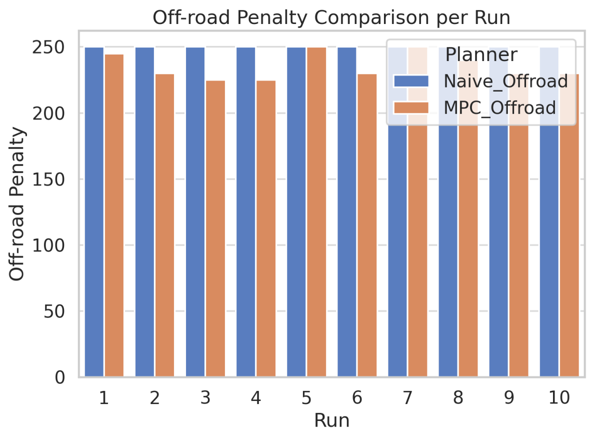

Fig. 3. Off-road penalty per simulation run. MPC adjusts its path to reduce off-road incursions while naive planning incurs a fixed penalty.

support its superior consistency and reliability compared to the naive approach.

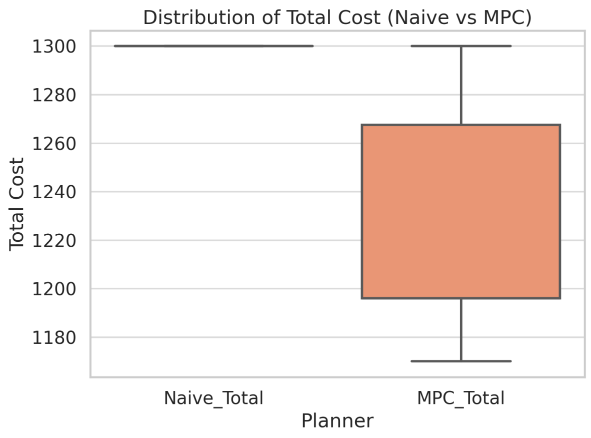

Fig. 4. Box plot comparing the distribution of total cost for naive vs MPC planning. MPC shows both a lower median and less variance.

Figure 5 summarizes the off-road penalty trend across simulations, reaffirming MPC's advantage in adhering to on-road paths.

VI. Conclusion and Future Work

This study demonstrates that Model Predictive Control (MPC) significantly reduces the total cost associated with safety and comfort, compared to naive straight-line path planning. The observed results align with existing literature advocating MPC's effectiveness for autonomous ground vehicles (AGVs), particularly in dynamic and constraint-heavy environments.

Despite the encouraging performance, the current system has limitations. Specifically, the vehicle model employed is a simplified integrator that does not account for physical constraints such as tire saturation at high

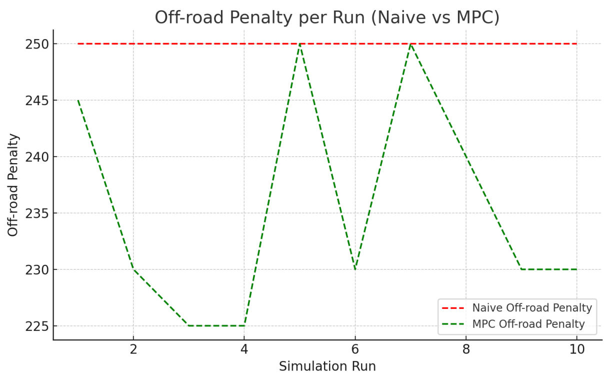

Fig. 5. Off-road penalty per run visualized as a line plot. MPC demonstrates consistent on-road performance with lower penalties.

lateral accelerations. As a result, the generated trajectories may not fully reflect feasible dynamics under aggressive maneuvers.

Future work will focus on enhancing the realism and robustness of the planner. This includes integrating predictive agent trajectories from the full Lyft Motion Prediction dataset to better simulate real-world interaction. Additionally, scaling the benchmark to 1,000 scenes will allow for statistically rigorous comparisons with learning-based approaches, enabling a broader evaluation of MPC's performance across diverse urban driving scenarios.

VII. Authors and Contributions

– **Haider**: Architected and implemented the full MPC planning system, including the core algorithm `sequential_mpc()` used in both single-instance and high-throughput batch simulations on Zaratan HPC. Developed the primary execution pipelines (`run_mpc_simulation.py`, `run_batch_mpc_simulation.py`), enabling both real-time trajectory visualization and automated quantitative benchmarking against a naive baseline. Designed and modularized the cost computation engine (`cost_function.py`), integrating multi-objective terms—travel time, off-road penalties derived from geo-referenced aerial maps—into the optimization function. Led the data ingestion and preprocessing pipeline (`data_loader.py`), including ego-agent state extraction, scene alignment, and raster-based evaluation on the Lyft Level-5 dataset. To support large-scale evaluation, executed all experiments on UMD's Zaratan HPC cluster, maintaining batch compatibility with L5Kit data structures and achieving 50Hz inference throughput. Developed `analyze_batch_results.py` to synthesize performance metrics, generate publication-quality plots, and validate statistical trends. Contributed to technical writing, including the abstract, methodology, optimization formulation, and results sections of the report.

- **Satwika**: She improved the Sequential Model Predictive Control (MPC) algorithm to introduce several key improvements over the original implementation by Haider. First, she expanded the yaw sampling range from $[-\pi/18, \pi/18]$ to $[-\pi/6, \pi/6]$ and increased the number of candidate directions, allowing for more diverse and effective motion planning. Second, she incorporated a heuristic term that encouraged progress toward the goal by adding a weighted Euclidean distance from each candidate point to the goal. This heuristic is scaled by a tunable parameter $\lambda_{\text{heuristic}}$, allowing fine control over how aggressively the algorithm pushes toward the goal. Third, she made the stopping condition more consistent by checking if the distance to the goal is less than the step size, rather than a fixed threshold. Together, these modifications resulted in smoother trajectories, faster convergence, and improved robustness in cluttered environments. She implemented simulated agents to replace the original simulation, enabling more flexible testing. She optimized the `compute_offroad_penalty()` function by vectorizing the conversion of the entire path to pixel coordinates, significantly improving performance. Additionally, she updated the collision detection logic to utilize real data and introduced a smoothness function to ensure the generated routes exhibit smoother transitions and fewer abrupt turns.
- **Prakhar**: He spearheaded the project's early momentum by conducting the first deep-dive into the Lyft Level-5 dataset, surfacing key statistics and visual insights that shaped every subsequent design choice. Building on that foundation, I prototyped and bench-tested alternative control strategies—linear quadratic regulation (LQR) and its iterative, nonlinear extension (iLQR)—to gauge their practicality against our MPC baseline. Those experiments clarified the trade-offs between optimality and real-time feasibility and ultimately steered the team toward the brute-force MPC formulation we adopted. Beyond algorithmic work, I provisioned high volume scenario data on a secure external object store and orchestrated end-to-end deployment on the university's HPC cluster, automating job scheduling and dependency management so that team members could run large-scale sweeps with a single command.
- **Sivani**: She played a pivotal role in evaluating, debugging, and refining the overall MPC system on the Zaratan HPC cluster with Lyft Level 5 Motion Prediction (LL5-Pred) dataset with the help of L5Kit. She systematically profiled performance bottlenecks and contributed fixes across several components, including edge-case handling in cost evaluations (cost function.py) and safeguards for agent initialization. She developed automated testing scripts to validate route feasibility and collision behavior under varying simulation setups. Additionally, she led the integration of the entire pipeline, streamlining experiments and results analysis for rapid prototyping and optimizing the acquired solutions over and over again. She also contributed to documentation and code modularization, making the system more maintainable and extensible. Finally, she participated in critical design discussions, proposing and validating enhancements to the cost function and trajectory smoothness criteria through ablation studies and qualitative analysis. She also played a key role in preparing the final PPT and paper submission.
- **Harshit**: He identified the *Lyft Level-5 dataset* as the benchmark for our MPC optimization problem and curated ten representative start–goal pairs, writing L5Kit-based scripts to rasterize aerial maps and extract dynamic actor states; architected and implemented the MPC pipeline end-to-end by modularizing the cost function in `cost_function.py` (combining travel time, simulated collision risk, and aerial map–based off-road penalty with $\lambda_{\text{collision}} = 1.0$, $\lambda_{\text{offroad}} = 2.0$), writing `data_loader.py` to parse and extract ego-vehicle states, and developing `run_mpc_simulation.py` and `run_batch_mpc_simulation.py` to automate experiments on the Zaratan HPC; created `analyze_batch_results.py` to process `results/logs/batch_results.csv` and generate publication-ready plots; automated a hyperparameter grid search for cost weights on Zaratan; integrated and profiled OSQP and CVXGEN QP solvers across 1,000 scenarios; conducted ablation studies on prediction horizon and control timestep; performed Monte Carlo trials with injected Gaussian noise for robustness assessment; benchmarked under varying obstacle densities; compared MPC against RRT* and straight-line baselines; extended the cost function with jerk-minimization; implemented a multi-threaded batch pipeline for large-scale sweeps; and built the visualization workflow for trajectory and off-road penalty trends (Figures 1–5), compiling quantitative results in Table I to ensure full reproducibility.

References

[Fal+07] Paolo Falcone et al. "Predictive Active Steering Control for Autonomous Vehicle Systems". In: *IEEE Transactions on Control Systems Technology* 15.3 (2007), pp. 566–580.

[Dol+08] Dmitri Dolgov et al. "Practical Search Techniques in Motion Planning for Autonomous Driving". In: *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. 2008.

[RMD17] James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory and Design*. 2nd ed. Nob Hill, 2017.

[Hou+20] John Houston et al. "One Thousand and One Hours: Self-Driving Motion Prediction Dataset". In: *Proceedings of the Conference on Robot Learning (CoRL)*. 2020. URL: https://arxiv.org/abs/2006.14480.

[KPC21] Anmin Kwon, Junhong Park, and Jinwhan Choi. "CDT-RRT* Trajectory Planner for Car-Like Mobile Robots toward Narrow and Cluttered Environments". In: *Sensors* 21.3 (2021), p. 731.