SeerOpt: Bidirectional Learning and Optimization for Future-Aware Driving

Harsh Pandit[1], Kush Patel[1], and Kunal Siddhawar[1]

***Abstract*— We introduce a new methodology that aims to bridge the gap between flexible neural planning and reliable constraint driven optimization in autonomous driving. Building on the work of SeerDrive [14], our approach addresses the persistent problem that neural planners often violate essential safety and physical constraints, leading to infeasible or unsafe trajectories. To mitigate these issues we investigate two optimization methods, GTSAM based factor graph optimization and Nonlinear Model Predictive Control. We describe their formulation, integration with future bird eye view predictions, and their ability to refine neural trajectories. We present qualitative and quantitative comparisons that highlight the strengths and limitations of each method. Our results demonstrate that optimization based refinement can improve safety performance while preserving the benefits of future aware neural prediction. The complete source code and implementation details can be accessed here: `https://github.com/kushpatel19/SeerOpt`.**

I. INTRODUCTION

In recent years autonomous driving research has advanced rapidly through the use of neural network based systems that integrate perception, prediction, and planning. Although these models capture complex scene interactions, they frequently overlook the strict safety requirements needed for reliable deployment. Without explicit enforcement of physical and safety constraints, neural planners can generate trajectories that are infeasible, unstable, or prone to collisions.

This issue is evident in end to end models such as SeerDrive. SeerDrive predicts future bird eye view scenes together with planned motion and achieves strong performance on NAVSIM[3]. However, its predictions do not guarantee compliance with vehicle dynamics or temporal collision avoidance. In contrast, optimization based planners such as GTSAM [4], model predictive control (MPC)[13] enforce these constraints but depend on simplified representations and lack the expressiveness of neural models. This creates a persistent gap between flexible prediction and reliable safety.

This work addresses this gap by refining neural trajectories through an optimization process that introduces explicit safety constraints. The goal is to ensure that planned motion remains feasible and collision free across time by using future scene information predicted by the neural model. We show that this approach improves safety without retraining and reduces common failure modes of neural planning on NAVSIM.

[1]All four authors are from University of Michigan, Ann Arbor. `harshpan@umich.edu`, `kushkp@umich.edu`, `siddhawr@umich.edu`

II. RELATED WORK

Prior work uses factor-graph optimization (GPMP [11], King-Smith et al. [7]) for constrained, collision-free trajectory planning, but these approaches do not refine future-aware neural planners or exploit predicted BEV scene evolution as in our method. Hose et al. [5] propose approximate nonlinear MPC with safety-augmented neural networks, where a neural policy predicts an open-loop input sequence that is accepted only if it satisfies an MPC-derived feasibility and cost check, otherwise falling back to a safe candidate, yielding deterministic safety guarantees at much lower online compute cost. Their framework combines fast neural predictions with optimization-based safety for generic nonlinear MPC, whereas our work applies this idea to autonomous driving by refining SeerDrive's future-aware trajectories using predicted BEV maps.

III. METHODOLOGY

A. *SeerDrive Pipeline*

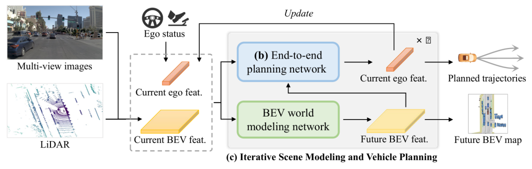

Fig. 1. SeerDrive Pipeline

Multi view images $\mathcal{I}$ and LiDAR features $\mathcal{P}$ are first fused to generate the current BEV feature map F_{bev}^{curr}, from which the current BEV semantic map $\mathcal{B}_{curr}$ is decoded. Anchored multi modal trajectories $\mathcal{T}$ and ego state $\mathcal{E}$ are encoded to obtain ego features F_{ego}^{curr}:

$$\begin{aligned} F_{\text{bev}}^{\text{curr}} &= \text{TransFuser}(\mathcal{I}, \mathcal{P}), \\ F_{\text{ego}}^{\text{curr}} &= \text{EgoEncoder}(\mathcal{T}, \mathcal{E}), \\ B_{\text{curr}} &= \text{BEVDecoder}(F_{\text{bev}}^{\text{curr}}). \end{aligned} \tag{1}$$

The overall architecture of SeerDrive is illustrated in Figure 1. A BEV world model then predicts future scene features F_{scene}^{fut} and corresponding future BEV features and maps:

$$\begin{aligned} F_{\text{scene}}^{\text{fut}} &= \text{BEVWorldModel}(F_{\text{scene}}^{\text{curr}}), \\ B_{\text{fut}} &= \text{BEVDecoder}(F_{\text{bev}}^{\text{fut}}). \end{aligned} \tag{2}$$

Current and future ego features are refined using transformer decoders with BEV features, and two sets of trajectories $\mathcal{T}_a$ and $\mathcal{T}_b$ are predicted:

$$
\begin{aligned}
F_{\text{ego}}^{\text{curr}} &= \text{TransformerDecoder}(F_{\text{ego}}^{\text{curr}},\ F_{\text{bev}}^{\text{curr}}), \\
F_{\text{ego}}^{\text{fut}} &= \text{TransformerDecoder}(F_{\text{ego}}^{\text{fut}},\ F_{\text{bev}}^{\text{fut}}), \\
\mathcal{T}_a &= \text{EgoDecoder}(F_{\text{ego}}^{\text{curr}}), \\
\mathcal{T}_b &= \text{EgoDecoder}(F_{\text{ego}}^{\text{fut}}).
\end{aligned} \tag{3}
$$

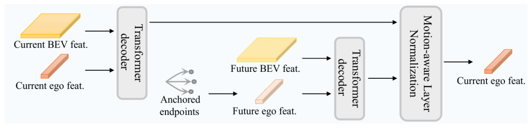

Fig. 2. Future aware End to end Planning Network

Future ego features are then merged into the current representation using motion aware layer normalization (MLN), and the final trajectory $\mathcal{T}_{final}$ is produced:

$$
\begin{aligned}
F_{\text{ego}}^{\text{curr}} &= \text{MLN}(F_{\text{ego}}^{\text{curr}},\ F_{\text{ego}}^{\text{fut}}), \\
\mathcal{T}_{\text{final}} &= \text{EgoDecoder}(F_{\text{ego}}^{\text{curr}}).
\end{aligned} \tag{4}
$$

Future BEV features also serve as a reference signal for trajectory refinement and are fed back into the world model to update future predictions. The future aware block, which refines trajectories using predicted future features, is highlighted in Figure 2.

Training consists of BEV semantic map loss $\mathcal{L}_{map}$ and trajectory loss $\mathcal{L}_{traj}$, including current and future map terms and N trajectory prediction iterations:

$$\mathcal{L}_{map} = \lambda_1 \mathcal{L}_{map}^{curr} + \lambda_2 \sum_{k=1}^{N} \mathcal{L}_{map}^{fut(k)}, \tag{5}$$

$$\mathcal{L}_{traj} = \lambda_3 \sum_{k=1}^{N} \left(\mathcal{L}_a^{(k)} + \mathcal{L}_b^{(k)} + \mathcal{L}_{final}^{(k)} \right), \tag{6}$$

$$\mathcal{L}_{total} = \mathcal{L}_{map} + \mathcal{L}_{traj}. \tag{7}$$

B. Environmental Constraint Extraction

To support geometric constraint enforcement, we extract drivable area polygons directly from the Bird Eye View semantic map. As illustrated in Figure 3, the road network is converted into a collection of Shapely polygons denoted as $\mathcal{P}_{drive} = \{P_1, P_2, \ldots, P_M\}$. These polygons represent lane regions, intersections, and connectors, providing a precise vector based description of the navigable space.

This representation allows efficient signed distance evaluation during optimization. For any candidate state x_k, a penalty is applied when it falls outside the union of polygons $\bigcup \mathcal{P}_{drive}$. This discourages trajectories that leave the drivable region and ensures that both GTSAM and NMPC refinement operate within valid road boundaries.

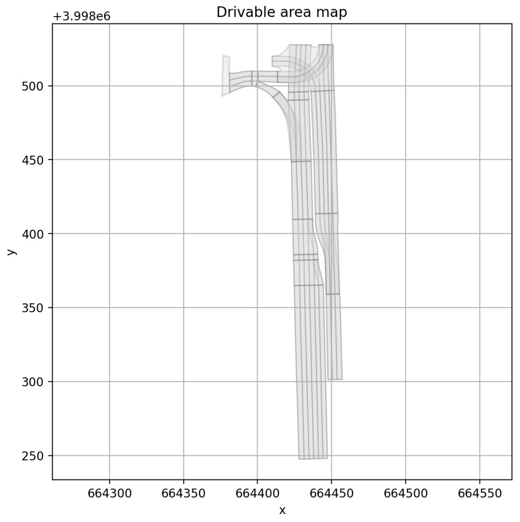

Fig. 3. Visualization of the extracted drivable area polygons

C. GTSAM Trajectory Optimization

While the SeerDrive baseline provides a strong initial trajectory prediction, purely learning-based planners can occasionally produce kinematically infeasible paths or erratic control actions. To address this, we introduce a post-processing optimization layer using the Georgia Tech Smoothing and Mapping (GTSAM) library. This layer refines the raw SeerDrive trajectory by formulating the planning problem as a factor graph optimization [6], enforcing smoothness and consistency while respecting the planner's original intent.

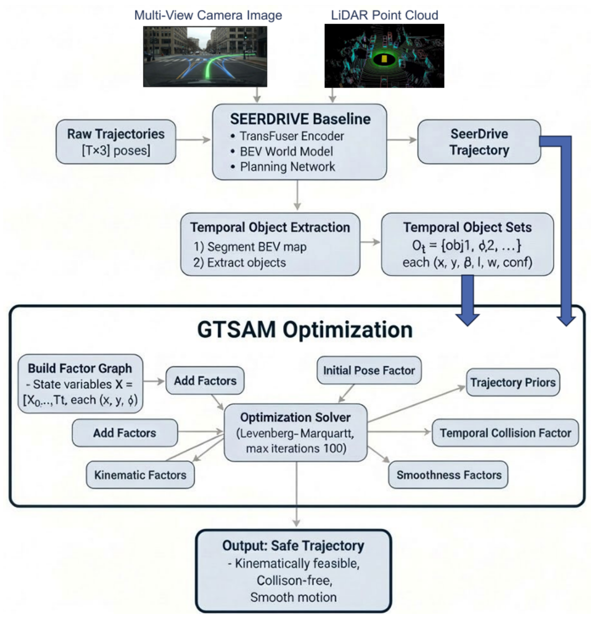

Fig. 4. The SeerOpt pipeline integrating GTSAM Optimization

1) Factor Graph Formulation: We represent the vehicle's trajectory as a set of discrete states $X = \{x_0, x_1, \ldots, x_N\}$, where each state $x_k \in \mathbb{R}^2$ represents the vehicle's position

(x, y) in the local coordinate frame at time step k. The optimization seeks to find the maximum a posteriori (MAP) estimate of the trajectory by minimizing a nonlinear least-squares error function defined by three key factors [8]:

$$X^* = \arg\min_X \Bigg(\mathcal{E}_{init} + \sum_{k=0}^{N} \mathcal{E}_{prior}(x_k) + \sum_{k=0}^{N-1} \mathcal{E}_{smooth}(x_k, x_{k+1}) \Bigg) \tag{8}$$

Initial Pose Factor ($\mathcal{E}_{init}$): To ensure continuity with the vehicle's current state, we anchor the first point of the trajectory x_0 to the ego-vehicle's current position derived from the SeerDrive prediction start point. This is modeled as a unary factor with a strict noise model (Σ_{init}):

$$\mathcal{E}_{init} = \|x_0 - x_{start}\|^2_{\Sigma_{init}} \tag{9}$$

Trajectory Prior Factor ($\mathcal{E}_{prior}$): This factor keeps the optimized trajectory close to the original SeerDrive prediction (x_k^{seer}). It ensures that the optimization does not deviate significantly from the learned planner's semantic intent (e.g., lane keeping or turn intent).

$$\mathcal{E}_{prior}(x_k) = \|x_k - x_k^{seer}\|^2_{\Sigma_{seer}} \tag{10}$$

where Σ_{seer} is a diagonal covariance matrix representing our confidence in the learning-based prediction.

Smoothness Factor ($\mathcal{E}_{smooth}$): To enforce kinematic feasibility and ride comfort, we minimize the difference between consecutive states, effectively penalizing high-frequency oscillations (jerky movements). This acts as a constant velocity prior between time steps [1]:

$$\mathcal{E}_{smooth}(x_k, x_{k+1}) = \|x_{k+1} - x_k\|^2_{\Sigma_{vel}} \tag{11}$$

2) Optimization Solver: The factor graph is solved using the Levenberg-Marquardt algorithm [10], a standard iterative non-linear least squares solver. We initialize the values with the raw SeerDrive trajectory points. The optimizer iteratively updates the state variables X to minimize the total error graph.

The resulting trajectory X^* represents a smoothed path that retains the global navigation intent of the SeerDrive model while mitigating local noise and kinematic inconsistencies. Finally, the optimized local trajectory is transformed into the global coordinate frame for valid metric evaluation and visualization against the ground truth and centerline.

D. Nonlinear Model Predictive Control (NMPC)

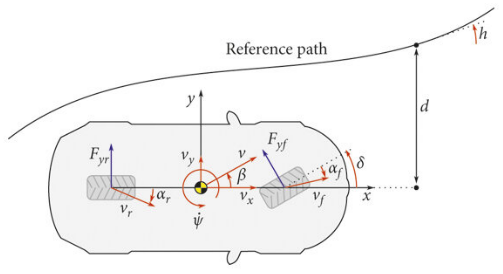

Fig. 5. Dynamic single track vehicle model

1) System Model and Dynamics: The vehicle state q, control inputs u, and auxiliary variables z are defined as:

$$q = \begin{bmatrix} U_x & U_y & r & x & y & \phi \end{bmatrix}, \tag{12}$$

$$u = \begin{bmatrix} F_x & \delta \end{bmatrix}, \tag{13}$$

$$z = \begin{bmatrix} F_{yf} & F_{yr} & z_{\mu f} & z_{\mu r} \end{bmatrix}. \tag{14}$$

where U_x, U_y are body velocities, r is yaw rate, (x, y) is global position, ϕ is heading, F_x is traction force, δ is steering angle, $F_{y_{f/r}}$ are lateral tire forces, and z_{μ_f} and z_{μ_r} denote intermediate slip-related variables.

The continuous-time vehicle dynamics are derived based on dynamic single-track vehicle model presented in [9].

2) NMPC Cost Function Design: SeerDrive provides a sparse 4 s horizon (0.5 s intervals), which proved insufficient for stable tracking via interpolation. Consequently, we incorporate only the initial and final SeerDrive states into the formulation to generate an optimal trajectory satisfying boundary conditions.

We minimize the tracking error $\mathcal{E}_q = q - q_{\text{ref,N}}$ by solving the following optimal control problem:

$$\begin{aligned} \min_{u(\cdot)} \quad & \sigma\big(q(t_H)\big) \;+\; \int_0^{t_H} l\big(q(t), u(t), t\big)\, dt \\ \text{subject to} \quad & q(0) = q_0, \\ & \dot{q}(t) = f(q(t)) + g(q(t), u(t)), \\ & h_{\text{eq}}(q(t), u(t), t) = 0, \\ & h_{\text{in}}(q(t), u(t), t) \geq 0. \end{aligned} \tag{15}$$

Here, t_H is the horizon. The running cost $l(q, u, t) = \mathcal{E}_q^T Q \mathcal{E}_q + uRu$ uses positive definite weighting matrices Q and R to balance tracking accuracy and control effort.

3) Constraints: We enforce friction-cone limits on tire forces [12], box constraints on states/inputs, power-limited traction force, and a minimum longitudinal velocity to prevent reversing:

$$\begin{aligned} F^2_{y_f,k} + F^2_{x_f,k} &\leq (\mu_f F_{z_f,k})^2 + z^2_{\mu_f}, \\ F^2_{y_r,k} + F^2_{x_r,k} &\leq (\mu_r F_{z_r,k})^2 + z^2_{\mu_r} \end{aligned} \tag{16}$$

$$q_{\min} \leq q \leq q_{\max}, \quad u_{\min} \leq u \leq u_{\max} \tag{17}$$

$$F_x \leq \frac{P_{\text{eng}}}{U_x}, \quad U_x \geq 2.0 \tag{18}$$

We formulate and solve the NMPC problem using `CasADi` [2]. The resulting nominal control inputs are applied to the vehicle dynamics model, simulated via a fixed-step Runge–Kutta (RK) integrator.

IV. Results and Discussion

As seen in Table I, both GTSAM and NMPC outperform the original SeerDrive trajectories in several metrics. However, there is still room for improvement. For example, the Comfort score for the NMPC-generated trajectories can be increased by adjusting the cost-function weights, particularly those penalizing aggressive steering and acceleration inputs. It is important to note that our current evaluation was performed on a representative subset of tokens, rather than the entire NAVSIM dataset.

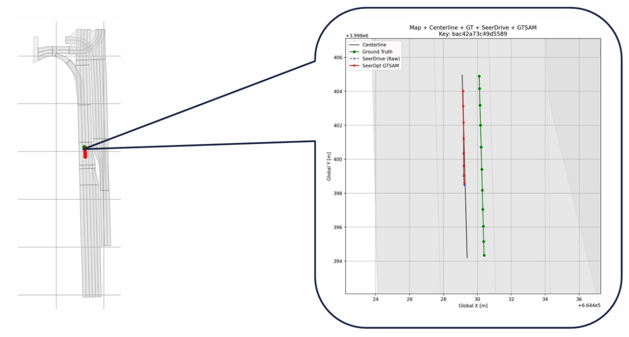

Fig. 6. GTSAM Optimized Trajectory vs SeerDrive Trajectory

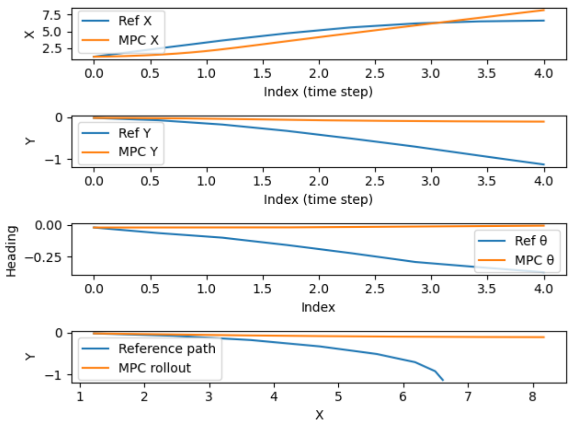

Fig. 7. NMPC vs SeerDrive Trajectory

The Fig.7 compares the NMPC trajectory with SeerDrive Trajectory. The NMPC controller produces 361 poses over the duration of 4s. The deviation may be attributed to the short duration of the reference trajectory provided by SeerDrive, which limits the predictive capability of the NMPC horizon. Another contributing factor could be the tuning of MPC weights and constraints, which may not fully capture the desired tracking behavior.

V. Conclusion

In conclusion, SeerOpt demonstrates the efficacy of a hybrid planning framework that augments future-aware neural predictions with rigorous geometric optimization. By integrating a post-processing layer using GTSAM factor graphs and nonlinear MPC, we successfully enforced kinematic feasibility and safety constraints that are often violated by purely learning-based models. Our evaluation on the NAVSIM benchmark highlights that trajectory refinement improves collision avoidance metrics while preserving the semantic intent of the original planner. These findings validate the potential of bidirectional learning-optimization pipelines as a robust solution for safety-critical autonomous driving systems.

VI. Future Works

While SeerOpt successfully demonstrates the benefits of combining neural planning with geometric optimization, there are several avenues for further improvement:

- **Current Ego-State Integration:** We plan to incorporate explicit dynamic constraints, such as velocity and steering limits, to ensure refined trajectories are strictly kinematically feasible.
- **Obstacle-Aware Optimization:** We aim to leverage predicted object motion from the world model as dynamic constraints in the factor graph, enabling proactive safety optimization in complex scenarios.
- **Real-Time Closed-Loop Control:** We propose combining GTSAM for global smoothing with Model Predictive Control (MPC) for local control to enable robust, real-time trajectory tracking.

VII. Acknowledgment

We would like to sincerely thank Prof. Maani Ghaffari for their invaluable guidance, insightful lectures, and continuous support throughout the course of this project. Their expertise in robotics and perception played a pivotal role in shaping our understanding and approach.

TABLE I
Comparison of Trajectory Performance Metrics

Type	NC	DAC	DDC	EP	TTC	Comfort	PDMS
SeerDrive	0.909	0.653	1	0.411	0.823	1	0.493965
GTSAM	0.910	0.684	1	0.411	0.841	1	0.517308
NMPC	0.908	0.760	1	0.319	0.827	0.758	0.485066

References

[1] Brian D. O. Anderson, John B. Moore, and Mansour Eslami. "Optimal Filtering". In: *IEEE Transactions on Systems, Man, and Cybernetics* 12 (1979), pp. 235–236. URL: https://api.semanticscholar.org/CorpusID:251583334.

[2] Joel A E Andersson et al. "CasADi: A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation*. Vol. 11. 1. Springer, 2019, pp. 1–36.

[3] Daniel Dauner, Marcel Hallgarten, Tianyu Li, et al. "NAVSIM: Data-Driven Non-Reactive Autonomous Vehicle Simulation and Benchmarking". In: *Neural Information Processing Systems*. 2024.

[4] Frank Dellaert. *Factor Graphs and GTSAM: A Hands-on Introduction*. Tech. rep. GT-RIM-CP&R-2012-002. Georgia Institute of Technology, 2012. URL: https://www.cc.gatech.edu/~dellaert/pubs/Dellaert12fnt.pdf.

[5] Henrik Hose et al. "Approximate Nonlinear Model Predictive Control With Safety-Augmented Neural Networks". In: *IEEE Transactions on Control Systems Technology* 33.6 (Nov. 2025), pp. 2490–2497. ISSN: 2374-0159. DOI: 10.1109/tcst.2025.3590268. URL: http://dx.doi.org/10.1109/TCST.2025.3590268.

[6] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. "iSAM: Incremental Smoothing and Mapping". In: *IEEE Transactions on Robotics* 24 (2008), pp. 1365–1378. URL: https://api.semanticscholar.org/CorpusID:1312251.

[7] Matthew King-Smith, Panagiotis Tsiotras, and Frank Dellaert. *Simultaneous Control and Trajectory Estimation for Collision Avoidance of Autonomous Robotic Spacecraft Systems*. 2022. arXiv: 2204.13251 [cs.RO]. URL: https://arxiv.org/abs/2204.13251.

[8] Rainer Kümmerle et al. "G2o: A general framework for graph optimization". In: *2011 IEEE International Conference on Robotics and Automation* (2011), pp. 3607–3613. URL: https://api.semanticscholar.org/CorpusID:206849534.

[9] Vincent A. Laurense and J. Christian Gerdes. "Long-Horizon Vehicle Motion Planning and Control Through Serially Cascaded Model Complexity". In: *IEEE Transactions on Control Systems Technology* 30 (2021), pp. 166–179. URL: https://api.semanticscholar.org/CorpusID:233804134.

[10] Jorge J. Moré. "The Levenberg-Marquardt algorithm: Implementation and theory". In: *Numerical Analysis*. Ed. by G. A. Watson. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 105–116. ISBN: 978-3-540-35972-2.

[11] Mustafa Mukadam et al. "Continuous-time Gaussian process motion planning via probabilistic inference". In: *The International Journal of Robotics Research* 37.11 (Sept. 2018), pp. 1319–1340. ISSN: 1741-3176. DOI: 10.1177/0278364918790369. URL: http://dx.doi.org/10.1177/0278364918790369.

[12] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[13] James Blake Rawlings and David Q Mayne. *Model predictive control: Theory and design*. Nob Hill Publishing Madison, WI, 2009.

[14] Bozhou Zhang et al. "Future-Aware End-to-End Driving: Bidirectional Modeling of Trajectory Planning and Scene Evolution". In: *Neural Information Processing Systems* (2025).