Path Planning on SE(2) for Autonomous Vehicles

Stefan Ramdhan
Computing and Software
McMaster University
Hamilton, Canada
ramdhans@mcmaster.ca

***Abstract*—Path planning for autonomous vehicles (AVs) is a critical step in the autonomy pipeline enabling AVs to plan their way through an environment toward a goal. Rapidly Exploring Random Trees (RRT) is a sampling-based method that constructs a graph of robot poses to find a path between the start and goal poses. This project combines the asymptotically optimal RRT* with Dubins Curves to generate a pose graph, followed by smooth and feasible path generation via interpolation on SE(2). A manifold-based optimization approach is investigated to find a smooth sequence of poses that obeys workspace constraints and non-holonomic constraints. The solution will be implemented using the CARLA [1] driving simulator, and evaluated using a forward-drive parking lot scenario.**

***Index Terms*—path planning, group theory, Lie groups, motion planning**

I. Introduction

Trajectory planning for non-holonomic robots has been a topic of interest to both industry and academia due to the application to Autonomous Vehicles (AVs), rovers, and even robot arms. To plan a trajectory, one must first plan a path, which is a sequence of points from an initial point to a final point, passing through pre-defined points [2]. A trajectory on the other hand is a path plus timing information [2]. This project focuses primarily on path planning, with trajectory planning as a stretch goal.

Planning paths for AVs requires adherence to many constraints, including static and dynamic obstacle avoidance, non-holonomic constraints, and performance considerations such as energy consumption and passenger comfort. A common approach to address these considerations is to split tasks between a global planner and a local planner. The global planner generates a rough, feasible, and collision-free path from a start pose to a goal pose. It must incorporate the non-holonomic constraints of the vehicle; otherwise it will produce paths that are infeasible for a vehicle to follow, such as requiring lateral velocity without longitudinal velocity.

Algorithms to plan paths are commonly categorized into grid-based methods and sampling-based methods. Grid-based methods discretize the planning space into cells, and produce paths by exploring these cells, according to a cost function. Examples include the A*, or its non-holonomic version Hybrid-A*. Sampling-based methods randomly explore free configuration space to build a tree of paths. Examples include Rapidly Exploring Random Trees (RRT), and Probabilistic Roadmaps (PRM). RRT is often used in practice because it scales better than A* as the state space increases in size, however it does not produce an optimal path [3]. Karaman and Frazzoli [3] introduced RRT* that produces asymptotically optimal paths without significant increases in computational complexity, by intermittently rewiring the tree based on a cost function. To incorporate non-holonomic constraints into the RRT* planner, Dubins [4] and Reeds-Shepp curves [5] are commonly used, which model simple vehicle motion using very basic motion primitives. Dubins curves are shortest paths between two poses for a simple vehicle that has a fixed forward velocity and a single turning radius, and only allows forward motion, while Reeds-Shepp curves use similar motion primitives, but also allow backward motion. This project will use RRT* with Dubins Curves as the global planner to generate a rough path to the goal pose.

RRT generates a series of waypoints that, if interpolated using piecewise-linear functions, will often generate jerky paths that contain unnecessary turns, and discontinuities at the waypoints [6]. A local planner is often used to determine how to get between waypoints smoothly, and perform local replanning in case dynamic obstacles such as other vehicles are blocking the path [7]. For simplicity, this project will only introduce static obstacles, but local planning is still required to determine the path taken between waypoints. This local planning step requires interpolation to generate a fine-grained, smooth path between these waypoints in a manner that respects the constraints imposed in the global planning stage. This project explores three such methods. The first method is interpolation by computing the geodesic on SE(2), successively between waypoints. The second method constructs an interpolating cubic spline on SE(2). Third, an optimization-based approach is explored to generate a path that is C^2 continuous, and respects kinematic constraints and obstacle constraints.

The rest of the report is organized as follows. Section II reviews the related literature on RRT* with non-holonomic constraints and interpolation on Lie groups. Section III formalizes the problem and outlines the three methods by which the sparse RRT* path will be interpolated. The experiments and results are given in Section IV. Finally, we conclude with Section V and discuss future work.

II. Related Work

There have been many studies focused on adapting RRT to embed motion constraints such as non-holonomy into the algorithm. Berenson *et al.* [8] adapted bi-directional RRT to handle robot constraints such as torque limits, and collision avoidance. Their adaptation, called Constrained Bi-directional RRT (CBiRRT), explores the dynamic configuration space, which changes based on what the robot arm is holding. RRT integrated with Adaptations of RRT that integrated Ackermann drive planning models such as the Dubins Curves and Reeds-Shepp curves are easily traversable, but far from optimal [9] [10]. Han *et al.* [11] use a vehicle model integrated with RRT, thus constraining the configuration space, and producing paths corresponding to parallel parking and other difficult parking scenarios. However, Han *et al.* does not use a constraint-aware path smoothing, thus violations to the encoded constraints in the final path could go unnoticed. Pepy *et al.* [12] also embed vehicle dynamics into the configuration space, however the final paths are suboptimal, but nonetheless feasible for a non-holonomic vehicle.

On the topic of interpolation on Lie groups, Pan *et al.* [6] smoothly interpolate between waypoints in a 3D configuration space by splitting the translation and rotation components of SE(3). The translation component is a vector in $\mathbb{R}^3$, so the authors construct a cubic B-spline through all positional components of the waypoints produced by RRT. Then, the rotational components in $SO(3)$ are splined through by taking the log map of the set of orientations, constructing a cubic B-spline in the Lie algebra $\mathfrak{so}(3)$, then applying the exponential map back onto $SO(3)$. This approach interpolates a C^2 continuous 3D curve, but does not incorporate kinematic constraints. To the best of my knowledge, there does not exist any literature on constraining a spline in the Lie algebra by kinematic constraints. Most Lie group interpolation techniques take an optimization-based approach because of the ability to incorporate constraints in the form of a cost function [6].

III. Problem Formulation & Methodology

A. Global Planning

We define the configuration space $\mathcal{C}$ of the AV as $SE(2)$. SE(2) is the Lie group representing the group of rigid transformations in 2D space. If $\mathbf{R} \in SO(2), \mathbf{t} \in \mathbb{R}^2$, we define a pose in 2D space to be $\mathbf{T} \in SE(2)$:

$$\mathbf{T} = \left(\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline 0 & 1 \end{array}\right) \in SE(2).$$

We partition $\mathcal{C}$ into free configuration space $\mathcal{C}_{\text{free}}$ and obstacle space $\mathcal{C}_{\text{obs}}$ such that,

$$\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{obs}} = \mathcal{C}.$$

Free configuration space represents space that the vehicle can occupy, while obstacle space represents any space that the vehicle cannot occupy, such as lane markings and unpaved roads. When RRT* samples randomly to build a tree of poses, it is limited to sampling within $\mathcal{C}_{\text{free}}$ and ensures that the entire path between one node and another is in $\mathcal{C}_{\text{free}}$. To account for the size of the vehicle, $\mathcal{C}_{\text{obs}}$ is inflated so that the state space only needs to include position and orientation. The state space $\mathbf{x}$ is given by $\mathbf{x} = [x, y, \theta]^T$.

Traditional RRT* must minimize a distance metric, which is typically Euclidean distance. For this project, RRT* with Dubins Curves will be used, so the distance metric will be the path length of the Dubins Curve.

RRT* generates a discrete, jerky path $\gamma_{RRT*} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{N-1}\} \subset SE(2)$. While RRT* is asymptotically optimal, often, even after thousands of nodes are generated, the path is jagged due to the random nature of RRT. In most motion planning implementations, paths that contain many sharp or jagged turns are typically smoothed, as the vehicle may need to slow down to execute these sharp turns, thereby expending more energy [13].

B. Interpolation

1) Piecewise Geodesic Interpolation: The first approach to interpolating on SE(2) is to compute the geodesic between poses, then combine them in a piecewise fashion. Geodesic interpolation on SE(2) is analogous to geodesic interpolation in $\mathbb{R}^3$. Simply take the log map of the difference in poses to represent the twist in the Lie algebra. This twist represents the change in pose required to go from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$. Since the Lie algebra is a linear vector space, we can interpolate between Lie algebra elements $\xi_i = [\rho_{1,i}\, \rho_{2,i}\, \omega_i]^T$ just as we would interpolate between vectors in $\mathbb{R}^3$. Then, we apply the exponential map to get back onto the Lie group.

The geodesic between two consecutive poses $\mathbf{x}_i, \mathbf{x}_{i+1} \in \gamma_{RRT*}$ is given by,

$$\gamma(t; \mathbf{x}_i, \mathbf{x}_{i+1}) = \mathbf{x}_i \exp(t \log(\mathbf{x}_i^{-1}\mathbf{x}_{i+1})), \quad t \in [0, 1].$$

To extend this approach to multiple successive poses, we can instead parameterize using $t \in [0, N-1]$ where N is the number of poses in γ_{RRT*}. The combination of these geodesics will create a piecewise geodesic interpolating function γ_{pgi}, which is an interpolating path between start and goal poses, passing through waypoints in γ_{RRT*} and is given by,

$$\gamma_{pgi} = \begin{cases} \mathbf{x}_0 \exp\left(t \log(\mathbf{x}_0^{-1}\mathbf{x}_1)\right), & t \in [0,1], \\ \mathbf{x}_1 \exp\left(t \log(\mathbf{x}_1^{-1}\mathbf{x}_2)\right), & t \in [1,2], \\ \vdots & \vdots \\ \mathbf{x}_{N-2} \exp\left(t \log(\mathbf{x}_{N-2}^{-1}\mathbf{x}_{N-1})\right), & t \in [N-2, N-1]. \end{cases}$$

Because piecewise geodesic interpolation does not encode non-holonomic constraints and obstacle constraints, this path will not be feasible for an AV. Furthermore, this approach is expected to contain discontinuities at the waypoints defined by RRT*. Within piecewise geodesic interpolation, each successive segment of γ_{pgi} is unaware of the first and second derivatives of the previous segment, which will result in discontinuities in function velocity and acceleration at the waypoints. To account for first and second derivatives at the waypoints, we can use a cubic spline between RRT* waypoints, which guarantees global C^2 continuity.

2) Cubic Spline Interpolation: A cubic spline on SE(2) can be computed by constructing a cubic spline in the Lie algebra $\mathfrak{se}(2)$, then applying the exponential map to the Lie group SE(2) [14]. Simply take the poses $\mathbf{x} \in \gamma_{RRT*}$, and apply a transformation to represent poses relative to reference pose $\mathbf{x}_0$. Next, take the log map to represent poses as a twist in the Lie algebra, then spline in the Lie algebra, and finally apply the exponential map to get back onto the Lie group.

Formally, take poses $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{N-1} \in \gamma_{RRT*}$. Transform global poses into local poses by a transformation relative to reference pose $\mathbf{x}_0$,

$$\Delta\mathbf{x}_i = \mathbf{x}_0^{-1}\mathbf{x}_i.$$

Then, take the log map of the relative poses:

$$\xi_i^\wedge = \log(\Delta\mathbf{x}_i) = \begin{bmatrix} 0 & -\omega_i & \rho_{1,i} \\ \omega_i & 0 & \rho_{2,i} \\ 0 & 0 & 0 \end{bmatrix} \in \mathfrak{se}(2).$$

Then, represent the element of the Lie algebra in its parameterized form,

$$(\xi_i^\wedge)^\vee = \xi_i = \begin{bmatrix} \rho_{1,i} \\ \rho_{2,i} \\ \omega_i \end{bmatrix} \in \mathbb{R}^3.$$

Then, spline between these vectors to produce a C^2 continuous spline $s(t) \in \mathbb{R}^3$. Finally, apply the exponential map to get back onto the Lie group, then de-reference the poses back to the global frame:

$$\gamma_{sp} = \mathbf{x}_0 \exp(s(t)).$$

This path γ_{sp} will be C^2 continuous, however it will not be feasible for an AV because it is not constraint-aware. This method does not encode a kinematic model of the vehicle and thus will not respect the non-holonomic constraints of the vehicle.

3) Optimization-based Interpolation: As previously mentioned, optimization is a standard approach to produce a path restricted by many constraints. In this subsection, we will formalize a Gauss-Newton optimization approach to produce a path γ_{GN} that adheres to all of the constraints we desire. As an initial guess, we will can use either the cubic spline interpolated path and the piecewise geodesic interpolated path, but realistically we would want to use an initial guess that is closer to the global optimum, which may be decided visually. First, we will define the residuals that compose the cost function. To ensure conformance with the kinematic constraints of the vehicle, we define a r^{kin} that ensures conformity with a kinematic bicycle model. The state space of the bicycle model is,

$$\mathbf{x} = [x, y, \theta]^T,$$

where position is given by the (x, y) tuple and orientation is given by θ. The control inputs are given by,

$$\mathbf{u} = [v, \delta]^T,$$

where v is the vehicle's velocity, and δ is the vehicle's steering angle.

We define r^{kin} as the difference between the actual pose update and the kinematic pose update. The discrete kinematic pose difference is given by,

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v}{L}\tan(\delta) \end{bmatrix},$$

where L is the length of the wheelbase. We can define the kinematic update in pose as,

$$\mathbf{x}_{k+1}^{\text{kin}} = \mathbf{x}_k + \dot{\mathbf{x}}_k\Delta t = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} v_k\cos(\theta_k) \\ v_k\sin(\theta_k) \\ \frac{v_k}{L}\tan(\delta_k) \end{bmatrix}\Delta t,$$

for $k = 0, 1, \ldots, K-1$.

We can then define the kinematic residual at index k as,

$$r_k^{\text{kin}} = \log((\mathbf{x}_{k+1}^{\text{kin}})^{-1}\mathbf{x}_{k+1}) = 0.$$

We require $r_k^{\text{kin}} = 0, \forall k$ because adherence to physics is a hard constraint.

Next, we will define the obstacle avoidance residual. We will define a Signed Distance Function (SDF) $d : \text{SE}(2) \to \mathbb{R}$, where $d(\mathbf{x}_k) > 0$ when $\mathbf{x}_k \in \mathcal{C}_{\text{free}}$ and $d(\mathbf{x}_k) \leq 0$ when $\mathbf{x}_k \in \mathcal{C}_{\text{obs}}$.

We will define our obstacle avoidance residual at step k as,

$$r_k^{\text{obs}} = \max(0, -d(\mathbf{x}_k)).$$

If a given pose is in $\mathcal{C}_{\text{free}}$, then $d(\mathbf{x}_k) > 0$, so $r_k^{\text{obs}} = 0$. However, if a pose is in $\mathcal{C}_{\text{obs}}$, then $d(\mathbf{x}_k) \leq 0$, so $r_k^{\text{obs}} = -d(\mathbf{x}_k) > 0$ and thus entry into $\mathcal{C}_{\text{obs}}$ will be penalized. We do not require a safety buffer, as $\mathcal{C}_{\text{obs}}$ is already inflated by the size of the vehicle plus a safety buffer.

Next, we will define the waypoint residual, to ensure that our final path passes exactly through the waypoints produced by RRT*. We define the waypoint indices as j, where the total number of waypoints in γ_{RRT*} is J. The stride s that must be applied to go from one RRT* waypoint to the next within the interpolated pose-path is given by $s = (K-1)/(J-1)$. We want the pose at index $k = sj$, $\mathbf{x}_{sj}$ to be exactly equal to the RRT* waypoint at index j, W_j.

$$r_j^{\text{wp}} = \log(\mathbf{x}_{sj}^{-1}W_j) = 0.$$

We must also define a residual that encourages C^2 continuity, because adherence to the kinematic model only requires C^0 continuity. We can do this by taking the second difference between poses and enforcing it to be 0, just as one would in cubic spline interpolation. The first forward finite-difference approximation is generally given by,

$$x'_k \approx \frac{x_{k+1} - x_k}{\Delta t},$$

and the second finite-difference approximation is given by,

$$x''_k \approx \frac{x_{k+1} - 2x_k + x_{k-1}}{\Delta t^2}.$$

The first forward finite-difference approximation between poses is then given by,

$$f'_k = \frac{\log(\mathbf{x}_k^{-1}\mathbf{x}_{k+1})}{\Delta t},$$

and the second finite-difference approximation between poses is given by,

$$f''_k = \frac{f'_{k+1} - 2f'_k + f'_{k-1}}{\Delta t}.$$

We can then define the C^2-continuity residual as,

$$r_k^{C^2} = \frac{f'_{k+1} - 2f'_k + f'_{k-1}}{\Delta t} = 0,$$

for $k = 1, 2, \dots, K-2$.

Finally, we define our actuator limits,

$$|\delta_k \leq \delta_{max}|, \qquad 0 \leq v_k \leq v_{max}.$$

The cost function J is defined as,

$$J(\mathbf{z}) = \sum_k \left\|r_k^{\text{kin}}\right\|^2 + \sum_j \left\|r_j^{\text{wp}}\right\|^2 + \sum_k [\max(0, -d(\mathbf{x}_k))]^2 + \sum_k \left\|r_k^{C^2}\right\|^2,$$

where $\mathbf{z} = [\mathbf{x}^T, \mathbf{u}^T]^T$, with

$$\mathbf{x} = [x_0, y_0, \theta_0, \dots, x_{K-1}, y_{K-1}, \theta_{K-1}]^T,$$
$$\mathbf{u} = [v_0, \delta_0, \dots, v_{K-1}, \delta_{K-1}]^T.$$

We then solve the nonlinear least-squares problem:

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \sum_{k=0}^{K-1} \|r_k^{\text{kin}}(\mathbf{z})\|^2 + \sum_{j=0}^{J-1} \|r_j^{\text{wp}}(\mathbf{z})\|^2 + \sum_{k=0}^{K-1} \left[\max(0, -d(X_k))\right]^2 + \sum_{k=1}^{K-2} \|r_k^{C^2}(\mathbf{z})\|^2. \tag{1}$$

C. Evaluation Methodology

This project will be implemented in two stages. First, the global path will be generated in Python using a modified version of Atsushi Sakai's RRT* Path Planner with Dubins Curves [15]. Then, the sparse RRT* path will be passed to a Python script that interfaces with the CARLA Python API [1], which generates the ego vehicle and simulates its interaction with the environment using a physics-based model. The solution will be evaluated in a forward-only parking lot scenario.

Evaluation metrics will aim to quantify the drive comfort and safety of the final path. In CARLA, this is done by placing an Inertial Measurement Unit (IMU) into the vehicle as would be done in a real vehicle. To assess drive comfort, lateral acceleration and lateral jerk are considered fundamental vehicle kinematic factors that shape passenger comfort [16]. Longitudinal acceleration and jerk are not used because the path is not time-parameterized; the vehicle maintains a constant velocity throughout the entire drive, thus longitudinal dynamics are not relevant for comfort in this evaluation. Finally, the number of collisions will be used as the primary metric to quantify safety.

The planner that will be used as a baseline for comparison will be the sparse, piecewise linearly interpolated path produced by RRT*, which is not SE(2) aware.

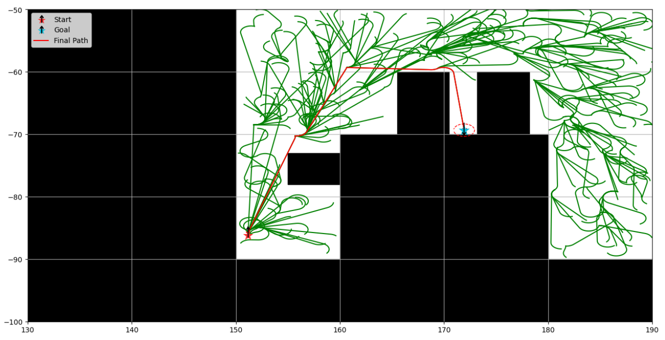


Fig. 1: Configuration space of the example scenario, with the random tree produced by RRT* in green. The final path produced by RRT* is in red.

IV. Experiments and Results

A bird's-eye-view of the example scenario is given in Figure 1, where the black space represents $\mathcal{C}_{\text{obs}}$, and the white space represents $\mathcal{C}_{\text{free}}$. The green lines indicate the paths that RRT* has sampled and the final path is indicated by the red line.

The poses generated by RRT* are sparse, as shown in Figure 2.

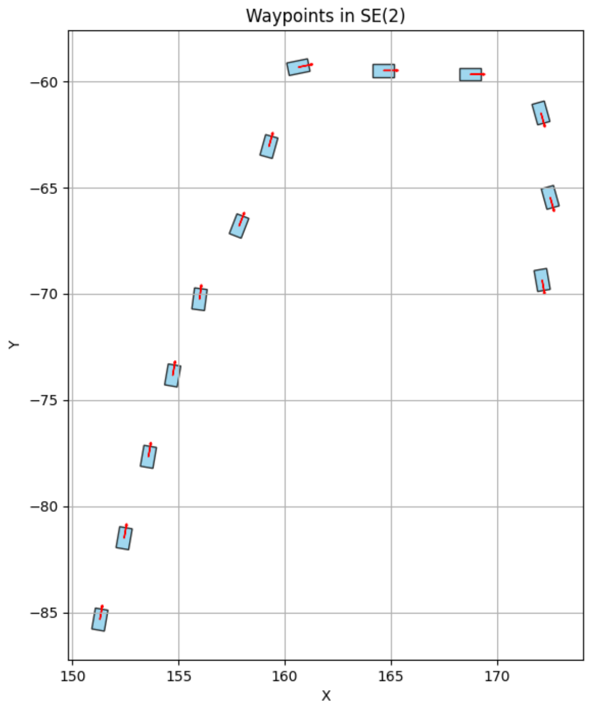


Fig. 2: RRT* generated pose graph.

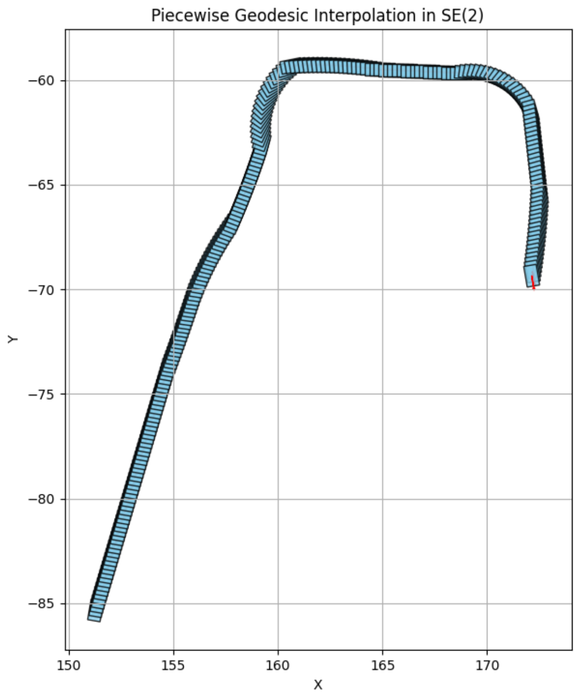


Fig. 3: Interpolated pose-path using Piecewise Geodesic Interpolation.

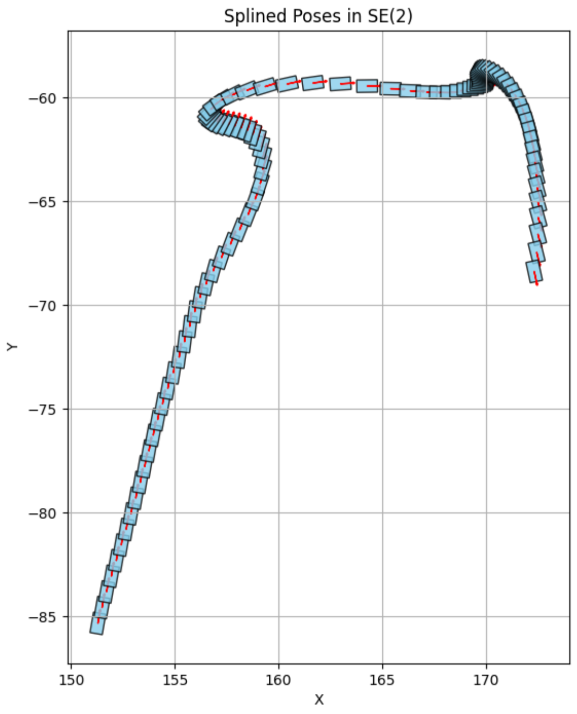


Fig. 4: Interpolated pose-path using Cubic Spline Interpolation.

Applying piecewise geodesic interpolation to γ_{RRT*} yields the path illustrated in Figure 3. The lack of C^1 continuity is most obvious at the waypoints. Visually, we can see that this path requires an instantaneous change in heading at certain points in the path. We can demonstrate C^0 continuity by plotting the interpolated function and its derivatives in the Lie algebra, which is given in Figure 5.

Applying cubic spline interpolation to γ_{RRT*} yields the path illustrated in Figure 4. Though this path is now C^2 continuous as depicted by Figure 6, this path is still not feasible for a vehicle with non-holonomic constraints, as it requires lateral velocity with little or no longitudinal velocity. This is expected, as a spline on SE(2) does not inherently incorporate a kinematic model of the vehicle and thus will violate the non-holonomic constraints. Furthermore, this interpolated path does not respect obstacle constraints, so there is a possibility that the vehicle will enter $\mathcal{C}_{\text{obs}}$, meaning that the vehicle may crash.

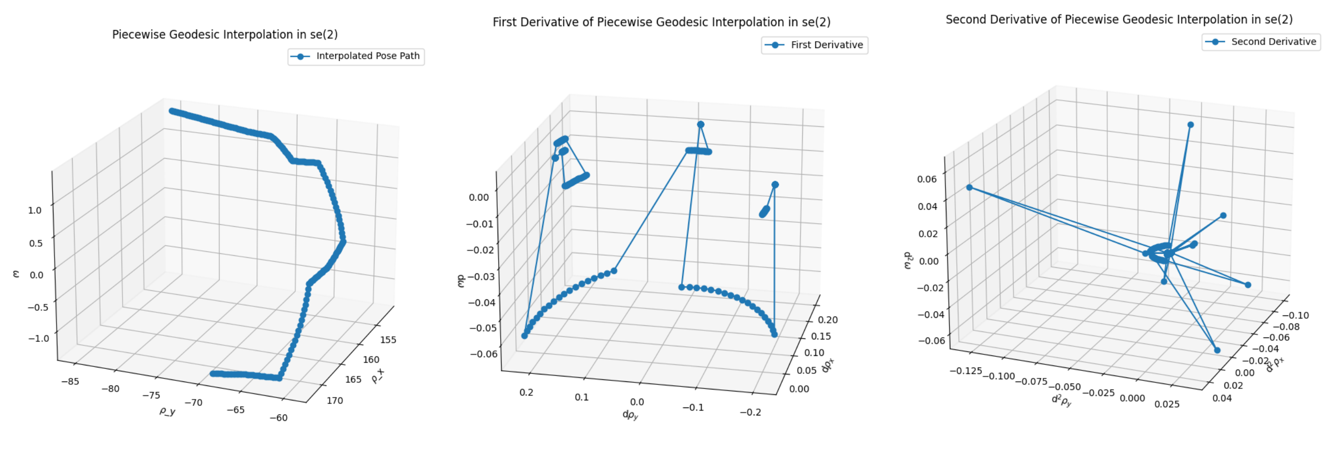
The top subplot in Figure 7 depicts the lateral acceleration on the vehicle using piecewise geodesic interpolation in blue, cubic spline interpolation in green, and SE(2)-unaware linear interpolation in red. At every waypoint before the sharp turn occurs at the 12 second mark, the linearly interpolated path incurs spikes in lateral acceleration, while piecewise geodesic interpolation and cubic spline interpolation smooth it out. Notably, piecewise geodesic interpolation has very large spikes in jerk during the sharp turn, indicating that C^0 continuity translates directly into uncomfortable driving. We cannot observe the C^2 continuity of the cubic spline smoothing out the path, because the cubic spline interpolated path causes the vehicle to crash into an obstacle just after the 15 second mark, demonstrated by the large spike in lateral acceleration and jerk. The subsequent low magnitude in lateral acceleration and jerk is due to the vehicle remaining in the position it crashed in for a very long time, attempting to continue to the next waypoint.

Summary statistics are given in Table I. Piecewise geodesic interpolation only increases the mean absolute lateral acceleration by 8.67%, but decreases mean absolute lateral jerk by 20.37%. A decrease in mean lateral jerk indicates that the piecewise geodesic interpolation is more predictable in terms of the change in lateral g-force a passenger would feel throughout the course of a ride, which is a desirable attribute. Cubic spline interpolation only superficially decreases mean average lateral acceleration and jerk, because it remains stationary due to many collisions with nearby vehicles for the majority of the scenario. This is also the reason for the lengthy drive time.

Summarized by Table II, the SE(2)-unaware linearly interpolated path and the piecewise geodesic interpolated path result in 0 crashes, the cubic spline interpolated path results in

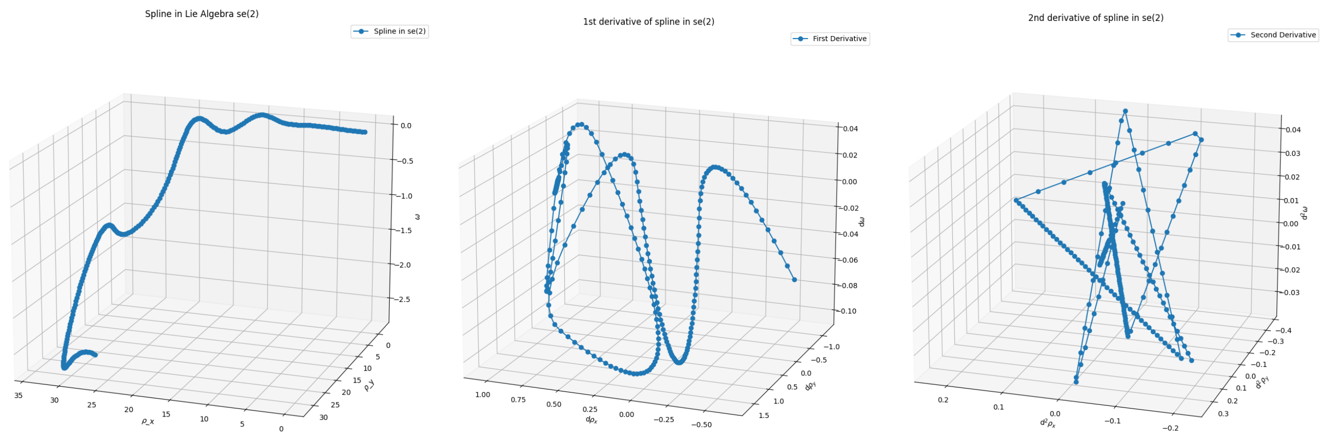
Method	**Mean Accel**$(\frac{m}{s^2})$	**Mean Jerk**$(\frac{m}{s^3})$	**Drive Time** (s)
Piecewise Geodesic	0.6126	3.1217	21.65
Cubic Spline	0.4117	2.3431	95.50
SE(2) Unaware Linear	0.5637	3.9205	20.55

TABLE I: Summary of Lateral Acceleration, Jerk, and Drive Time



(a) (b) (c)

Fig. 5: (a) Piecewise Geodesic Interpolation in $\mathfrak{se}(2)$. (b) First derivative of the interpolating function in $\mathfrak{se}(2)$. (c) Second derivative of the interpolating function in $\mathfrak{se}(2)$.



(a) (b) (c)

Fig. 6: (a) Cubic Spline Interpolation in $\mathfrak{se}(2)$. (b) First derivative of the interpolating spline in $\mathfrak{se}(2)$. (c) Second derivative of the interpolating spline in $\mathfrak{se}(2)$.

a total of 3 crashes. Here, PGI is short for piecewise geodesic interpolation, and CSI is short for cubic spline interpolation. These results are not surprising judging by the interpolated pose-paths for each of the interpolation methods.

As of the submission of this report, the Gauss-Newton optimization method has converged but the resulting pose-path does not adhere to any of the constraints imposed by the residuals likely due to a bug or convergence to a poor local minimum. For this reason, the pose-path from the optimization-based approach is not displayed in this report.

	SE(2) Unaware Linear	PGI	CSI
Number of collisions	0	0	3

TABLE II: Number of collisions for each interpolation method.

As a result of this project, I learned a lot about Lie groups and their application to non-holonomic motion planning. At first, I had incorrectly assumed that producing a cubic spline on SE(2) would enforce the vehicle's non-holonomic constraints. I believed this because rotation and translation are coupled due to SE(2) being a semi-direct product of SO(2) and $\mathbb{R}^2$, so I assumed that translation on SE(2) would inherently be aligned with orientation. Now I realize that SE(2) does not inherently encode motion constraints, meaning that lateral-only movement with respect to orientation is allowed. To constrain movement on SE(2), one needs to explicitly incorporate a kinematic model, and it is most natural to do so using an optimization-based approach. While formulating the optimization problem, I gained a deeper understanding of how to draw analogues from standard math to math on Lie groups, especially when I was defining the kinematic residual r^{kin} and the C^2-continuity residual r^{C_2}. I have also learned

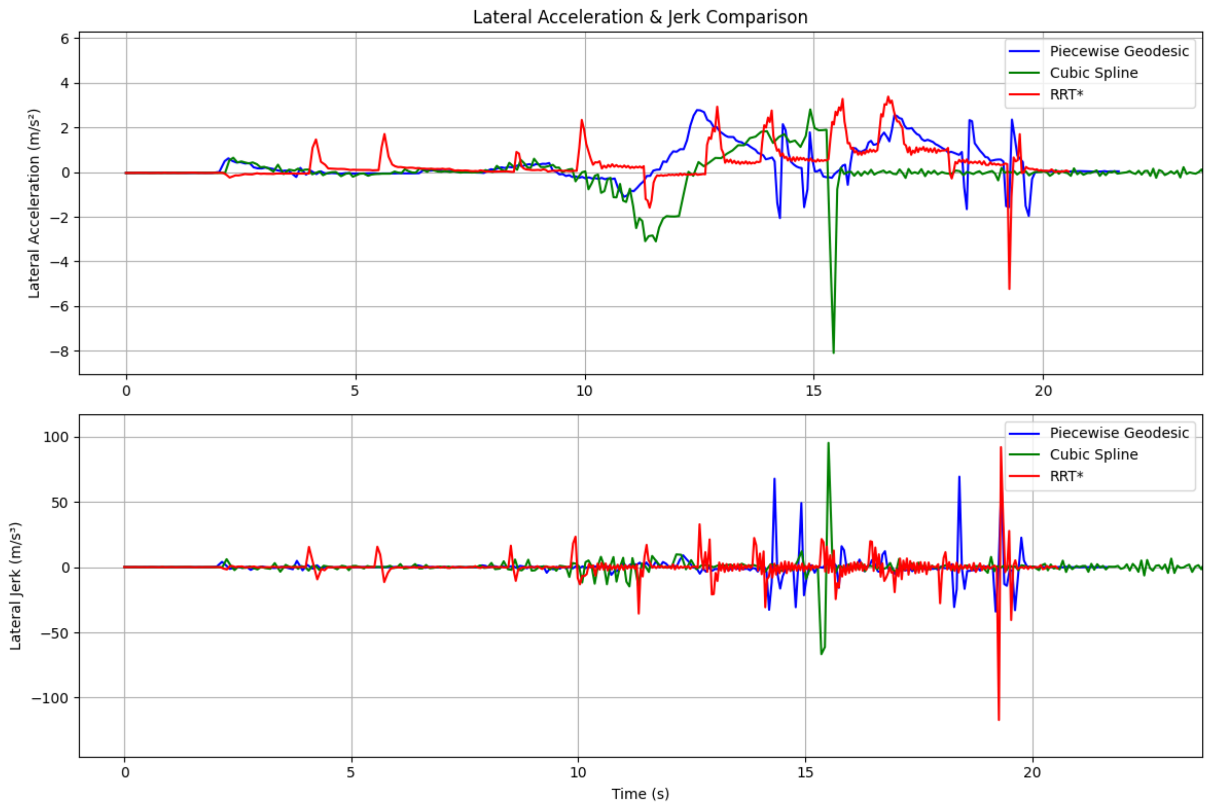


Fig. 7: Top: Lateral acceleration of the vehicle following its respective path. Bottom: Lateral acceleration of the vehicle following its respective path.

(and am still learning) about how to use and debug solvers such as those provided by SciPy [17] and CasADi [18].

V. Conclusion

This project demonstrated a two-stage path planner that combines an asymptotically-optimal global planner RRT* with Dubins Curves with three increasingly sophisticated Lie group-aware interpolation methods. Piecewise geodesic interpolation yielded a marginal increase in mean lateral acceleration but a sizeable decrease in mean lateral jerk, while maintaining zero collisions. The primary downside of piecewise geodesic interpolation is that it contains discontinuities at waypoints, meaning the interpolated pose-path is C^0 continuous. Piecewise geodesic interpolation also does not encode a kinematic model of the vehicle. For both reasons, the final path is not feasible for a vehicle. Cubic spline interpolation achieves global C^2 continuity, however its lack of kinematic and obstacle constraint-awareness produced a pose-path that was not feasible for a vehicle, resulting in three collisions. The optimization approach seems promising, however practical considerations have prevented the approach from producing a feasible path that obeys all of the constraints.

For future work, I will attempt to constrain a spline in the Lie algebra by kinematic and obstacle constraints, which I believe has not been done before. I will also investigate defining a subgroup of SE(2) in which movement on this subgroup is defined by a kinematic model, ensuring that splining in the Lie algebra respects that kinematic model. Finally, I will continue to work on the optimization approach to this problem, as I am very close to achieving the goal of feasible path generation subject to the constraints defined in Section III.

References

[1] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” 2017. [Online]. Available: https://arxiv.org/abs/1711.03938

[2] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, “Path planning and trajectory planning algorithms: A general overview,” in *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*, G. Carbone and F. Gomez-Bravo, Eds. Cham: Springer International Publishing, 2015, pp. 3–27.

[3] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” 2011. [Online]. Available: https://arxiv.org/abs/1105.1186

[4] L. E. Dubins, “On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents,” *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957. [Online]. Available: http://www.jstor.org/stable/2372560

[5] J. A. Reeds and L. A. Shepp, “Optimal paths for a car that goes both forwards and backwards,” *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[6] J. Pan, L. Zhang, and D. Manocha, “Collision-Free and Curvature-Continuous Path Smoothing in Cluttered Environments,” in *Robotics: Science and Systems VII*. The MIT Press, Jun. 2012, _eprint: https://direct.mit.edu/book/chapter-pdf/2269950/9780262305969_ccv.pdf. [Online]. Available: https://doi.org/10.7551/mitpress/9481.003.0035

[7] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, “Path planning techniques for mobile robots: Review and prospect,” *Expert Systems with Applications*, vol. 227, p. 120254, Oct. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742300756X

[8] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, “Manipulation planning on constraint manifolds,” in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 625–632.

[9] A. Khanal, “Rrt and rrt* using vehicle dynamics,” 2022. [Online]. Available: https://arxiv.org/abs/2206.10533

[10] A. Choudhary, Y. Kobayashi, F. J. Arjonilla, S. Nagasaka, and M. Koike, “Evaluation of mapping and path planning for non-holonomic mobile robot navigation in narrow pathway for agricultural application,” in *2021 IEEE/SICE International Symposium on System Integration (SII)*, 2021, pp. 17–22.

[11] L. Han, Q. H. Do, and S. Mita, “Unified path planner for parking an autonomous vehicle based on rrt,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 5622–5627.

[12] R. Pepy, A. Lambert, and H. Mounier, “Path planning using a dynamic vehicle model,” in *2006 2nd International Conference on Information and Communication Technologies*, vol. 1, 2006, pp. 781–786.

[13] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino, and C.-C. Peng, “Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges,” *Sensors*, vol. 18, no. 9, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/9/3170

[14] C. Tomlin, “Splining on lie groups,” University of California and Berkeley, Tech. Rep., 1995.

[15] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, “Pythonrobotics: a python code collection of robotics algorithms,” 2018. [Online]. Available: https://arxiv.org/abs/1808.10703

[16] C. Peng, C. Wei, A. Solernou, M. Hagenzieker, and N. Merat, “User comfort and naturalness of automated driving: The effect of vehicle kinematic and proxemic factors on subjective response,” *Applied Ergonomics*, vol. 122, p. 104397, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003687024001741

[17] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.

[18] J. Andersson, J. Gillis, G. Horn, J. Rawlings, and M. Diehl, “Casadi: a software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, vol. 11, 07 2018.