DENSE GRAPH NETWORK BASED PATH PLANNING ALGORITHM WITH GEOMETRIC RACELINE OPTIMIZATION

RT15E: PART REPORT

Lukasz Michalski*
Department of Software
PWR Racing Team
Poland, Wrocław, Sopocka 16
lukasz.michalski.pwrrt@gmail.com

ABSTRACT

Autonomous vehicle systems for Formula Student competitions require robust and efficient path planning algorithms capable of handling dynamic environments while ensuring kinematic feasibility and time-optimality. Traditional methods, such as Delaunay Triangulation & A*, Random Trees, Deep Reinforcement Learning, and Motion Planning with Trajectory Optimization, often fall short due to issues like path noise, suboptimal path selection, and computational inefficiency. This paper presents *Dense Graph based Path Planning Algorithm with Geometric Raceline Optimization*, that integrates a dense graph network with geometric raceline optimization to address these challenges. The algorithm operates in three phases: exploration, exploitation, and optimization. In the exploration phase, the search space is discretized using lane detection and kd-trees, while the exploitation phase refines the path through denoising and triangulation techniques. The optimization phase employs a spatiotemporal graph network to capture past and future vehicle states, enabling the calculation of a time-optimal racing line. Approach demonstrates its effectiveness through simulations and real-world tests, generating paths that closely approximate the ideal racing line, even in scenarios with limited environmental data or colorblind perception. By combining computational efficiency with robust path planning, algorithm offers a significant advancement for autonomous racing applications, paving the way for more reliable and competitive autonomous vehicles in Formula Student and beyond.

1 Introduction & Main Concept

The presented algorithm is an outcome of extensive testing conducted by our team over recent years in the development of an autonomous vehicle system. Rooted in the exploration of four fundamental algorithms—namely, *Delaunay Triangulation & A*, Random Trees, Deep Learning* and *Motion Planning with Trajectory Optimization* each approach, when implemented independently, exhibits certain limitations.

*Delaunay Triangulation & A** Kuruvilla [2022] search yields a path consistently traversing the center of the track. Moreover, the fine-tuning of the heuristic function for A* becomes non-trivial in the absence of cone color information.

While *Random Trees* are widely employed in the robotic industry for pathfinding, in the context of Formula Student, issues arise due to kinematic feasibility concerns at high speeds, leading to potential vehicle failure owing to random behavior and path shape noise.

The exploration of *Deep Reinforcement Learning* Bojarski et al. [2016] showcased promising results in simulations; however, challenges related to knowledge transfer, model maintainability for diverse dynamic events, and debugging issues during real-world tests rendered the approach impractical for our team.

An alternative approach, inspired by *Motion Planning with Trajectory Optimization* techniques Xu et al. [2012] McNaughton et al. [2011], ensures kinematically feasible and time-optimal paths. Unfortunately, this method demands

*RT15e, RT14e *Autonomous System Engineer*, RT13e *Path Planning Engineer*.

well-defined boundaries, and false positive detections from perception systems or error propagation in SLAM can render the path suboptimal for the vehicle to follow.

In response to the identified weaknesses and challenges inherent in the prior investigations, the proposed *Dense Graph based Path Planning Algorithm with Geometric Raceline Optimization* algorithm has been developed to systematically address each of these issues.

Architecture Information flow in the path planning system is handled by multiple sub-systems (algorithms) that perform calculations necessary to provide an optimal path for our vehicle. As input module takes the SLAM map or Perception data, and provides a path to follow by the Vehicle Controls module. Algorithm execution could be divided into three major phases: *exploration, exploitation* and *optimization* (Fig. 1).

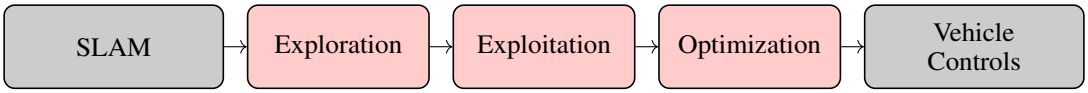

Figure 1: High-level path planning data flow diagram. Calculations in *Exploration* part are responsible for space discretization and map search. *Exploitation* part finds a generic path that is later refined in *Optimization* step.

2 Exploration

The path planning module functions on a landmark map sourced from either the Simultaneous Localization and Mapping (SLAM) module or observations collected by a Camera and LiDAR-based perception system in instances where SLAM functionality is disabled. While this configuration traditionally ensures adequate reliability concerning cone color information, its dependence is minimized to enhance the algorithm's resilience against potential camera failures and facilitate colorblind planning.

The Exploration phase undertakes the discretization of the search space through lane detection techniques, which approximate track geometry via the nearest neighbor approach and leverage kd-trees for computational efficiency. It is relevant to note that this stage is executed only when information regarding cone colors is available and estimations can be made. With this consideration, the search space is reduced to track boundaries, thereby enhancing the efficacy of subsequent stages within the exploration phase.

Following the space discretization process (track approximation has to be done on a full map), the SLAM map undergoes a filtration step. This filtration procedure eliminates cones positioned behind the vehicle using linear algebra techniques. By employing information such as heading, orthogonal vectors, rotation matrices, and a predefined distance threshold, cones beyond a 12-meter range and position behind the car are discarded. This approach optimizes the map search by focusing on landmarks within proximity to the vehicle and increases reliability by skipping distant landmarks with low accuracy and precision (Fig. 2).

Taking into consideration the potential track boundaries and the refined set of landmarks post-filtration, the crucial phase of map search is executed. Our chosen approach for this task is based on *rapidly exploring random trees* Tuncali and Fainekos [2019]. While this technique has traditionally been acclaimed as an industry standard for robot planning in indoor environments and grid maps, we have adapted it to suit our specific requirements, thereby rendering it an effective search technique for our purposes.

The Rapidly Exploring Random Trees (RRT) algorithm starts with the vehicle's current localization and gradually constructs a tree of potential paths by randomly exploring the configuration space. Typically, random trees aims to find a path from a starting point to a goal point. However, in our setup, the algorithm is tailored with specific parameters: a tree length threshold of 10 meters, a branch length of 1 meter, and an expansion angle of 20°. These adjustments ensure kinematic feasibility and reduce the computational cost. Given the track boundaries, the tree expansion is directed toward the estimated track direction penalizing the collision-detected directions (simple method without associated overhead Patil et al. [2021]), and prioritizing local exploration to detect sharp turns and generate longer paths. If track estimation is not feasible or if camera data is limited, an attention mechanism inspired by the pheromone concept from ACO Dorigo and Di Caro [1999] is introduced. This mechanism extends the stochastic sampling method by creating sampling zones with a radius of 3 meters around landmarks and constrained by 1 meter to avoid potential collisions, enhancing exploration efficiency. It is noteworthy to mention that this process is executed only on landmarks within a range exceeding 5 meters but constrained within 12 meters. Thresholds were manually designated based on perception accuracy.

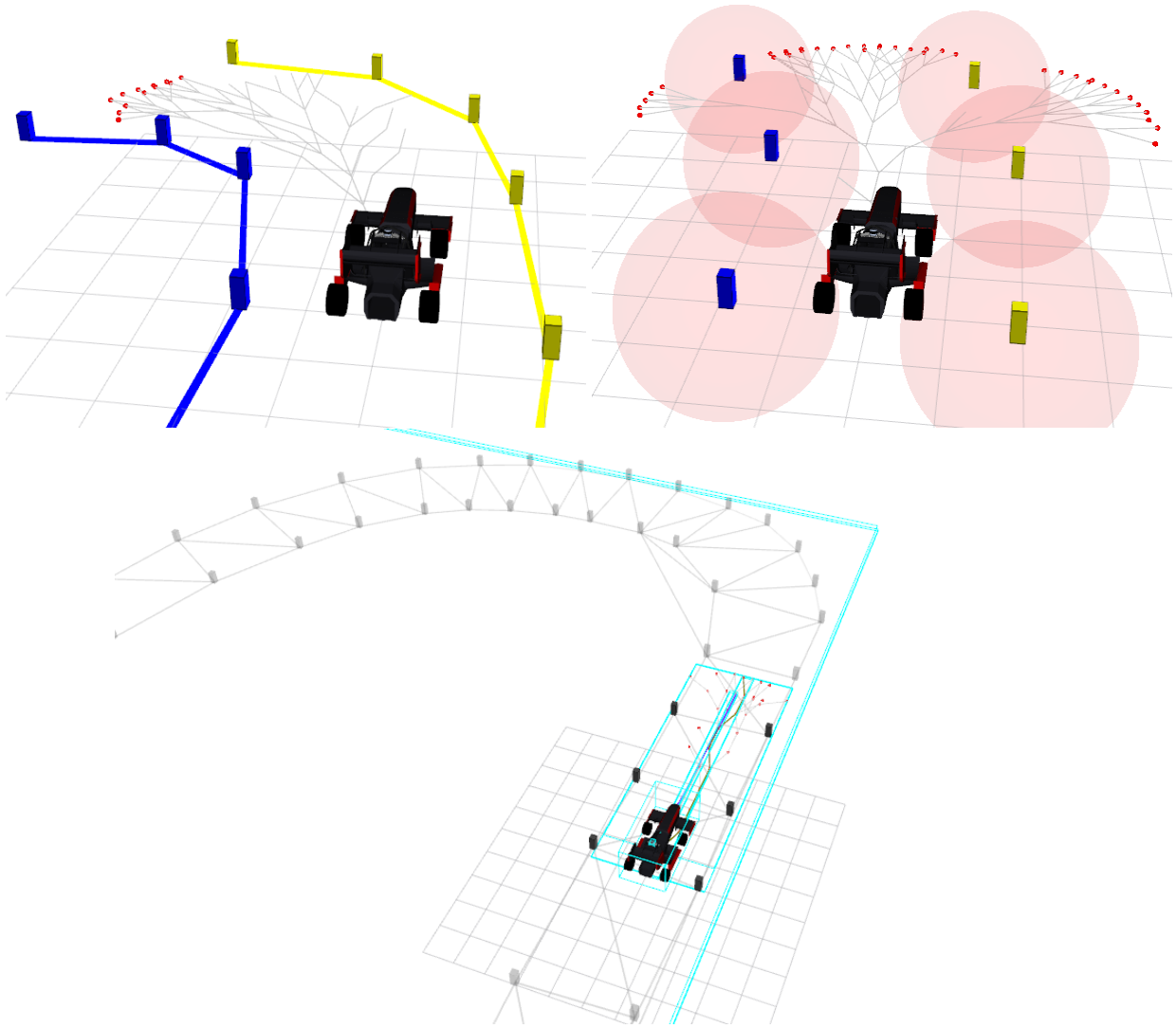

Figure 2: Approximated boundaries using nearest neighbor approach constraint tree expansion along the track. Attention zones are used to navigate map search mainly in colorblind planning or distorted and complex tracks with 180° turns

.

3 Exploitation

The exploitation phase is dedicated to identifying the optimal tree path, often characterized by significant noise and lacking kinematic feasibility. To address this, denoising techniques are employed to refine the path, smoothing its trajectory. Subsequently, line segment intersections are determined between the filtered tree path, and triangulation is performed on landmarks filtered during the exploration phase.

The selection of the best tree path is determined by a heuristic function operating on various weights. These include landmark cost and distance limit, favoring paths with cones on both sides without continuous gaps, distribution balance, and minimum path cost, which discards malformed paths (e.g., excessively short/sharp). In cases where such paths occur, the previously cached path is utilized.

Given the inherent randomness in the search approach, tree paths often lack kinematic feasibility. Hence, a denoising step is implemented to designate waypoints, operating with parameters such as the limit for distance changes along the path, weights for filtering new points, and a denoising counter to reset the state. The denoising process aims to smooth the path, reducing curvature and minimizing the randomness factor introduced by the random trees approach (Fig. 3).

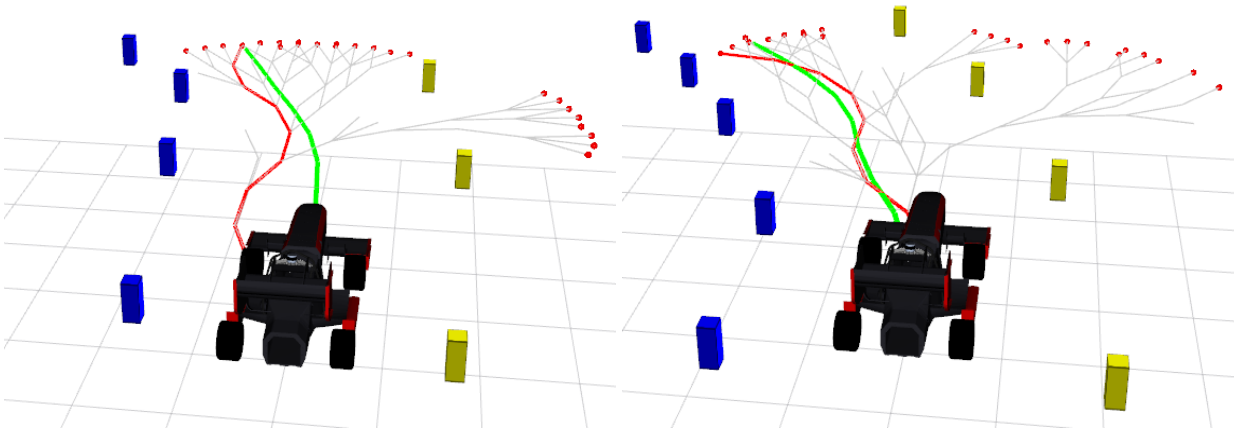

Figure 3: Denoising output (green path) comparison with best tree path (red path) from map search without track boundaries.

The resulting filtered path is then utilized for line segment intersection Qiu et al. [2013], thereby generating state space points along the track. Delaunay triangulation Jiang et al. [2010] is employed on the filtered landmark set to identify points along the track. Moreover, the triangulation is also filtered out by discarding the edges that may be not within the track (too long) or creating a triangle using three same colored landmarks (a scenario in sharp turns or detections from further track sectors). The denoised tree paths and triangulation edges are both considered in the line intersection procedure. Intersected points are considered waypoints and are sorted based on pose and intersection coordinates (Fig. 4).

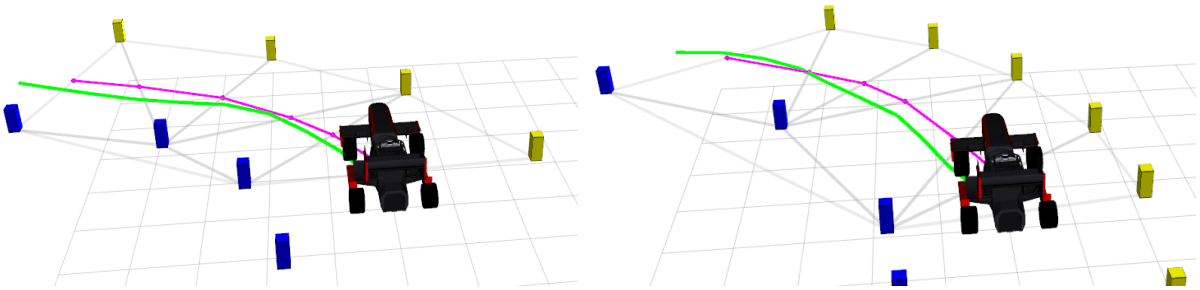

Figure 4: Best path (magenta) after triangulation and line intersection procedure (grey edges and green path).

Following the exploitation step, a generic path is established, facilitating navigation through the track. However, it is important to note that no optimization is performed to identify the racing line — an important factor when it comes to pushing the project to its limits.

4 Optimization

The primary challenge encountered in the Motion Planner with Trajectory Optimization lies in its reliance solely on the current state for the planning horizon, disregarding past decisions that may influence suboptimal path selections. To address this limitation, the introduction of the spatiotemporal graph network is proposed.

The fundamental concept underlying this approach involves capturing previously traversed track sectors with precise approximation and leveraging states generated by a lateral offset of a central waypoint. A custom cost function, dependent on vehicle forces during state-to-state transitions derived from the vehicle model and time estimation based on connection length and average velocity vector, is utilized to simulate activation for each layer (Fig. 6). The resulting output of the cost function is a numerical value associated with the edge weight in the graph.

It is noteworthy that connections between layers may be skipped if they overlap or are nearby, thereby incorporating states from the $t-1$ (Fig. 6) planning state alongside consecutive states, regularizing the resulting solution.

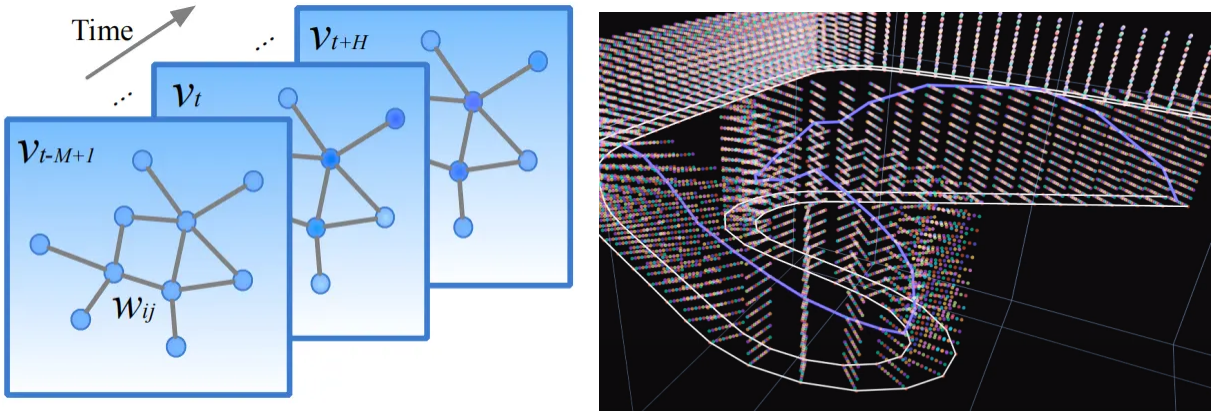

Figure 5: GraphNet builds upon multiple layers from past, current, and possible future states of the vehicle on the race track creating 3D Euclidean space for shortest path problem where each node in layer determines the velocity achievable by car.

Successive iterations expand the network with additional layers, increasing the graph's density and extending its representation into a 3-dimensional Euclidean space. Simultaneously, this process propagates past decisions that led to the current state, facilitating efficient tracking. By storing this information and establishing dense connections, it becomes possible to approximate curvature closely resembling real-world scenarios. Consequently, during subsequent laps and with proper loop closure without significant errors, the optimal path is acquired using either shortest path algorithms like Dijkstra/Bellman-Ford. In cases where the network becomes excessively large and algorithms for finding the optimal path violate real-time constraints, computational efficiency is enhanced through the utilization of metaheuristics - evolutionary algorithms, particle optimization, etc.

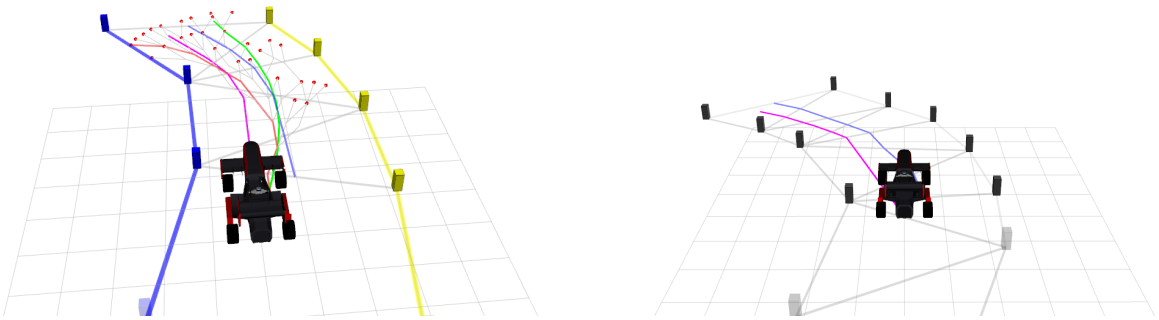

Figure 6: In magenta the race line calculated with *GraphNet* approach, in both colorblind (LiDAR sensor only perception system) and with extended environment information yields path which in simulations and real-world tests imitate the race line like curvature to follow.

References

Tanya Kuruvilla. Path planning for Formula Student driverless cars using Delaunay triangulation, Oct 2022. URL https://blogs.mathworks.com/student-lounge/2022/10/03/path-planning-for-formula-student-driverless-cars-using-delaunay-triangulation/.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. *CoRR*, abs/1604.07316, 2016. URL http://arxiv.org/abs/1604.07316.

Wenda Xu, Junqing Wei, John M. Dolan, Huijing Zhao, and Hongbin Zha. A Real-Time Motion Planner with Trajectory Optimization for Autonomous Vehicles. In *2012 IEEE International Conference on Robotics and Automation*, pages 2061–2067, 2012. doi:10.1109/ICRA.2012.6225063.

Matthew McNaughton, Chris Urmson, John M. Dolan, and Jin-Woo Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *2011 IEEE International Conference on Robotics and Automation*, pages 4889–4895, 2011. doi:10.1109/ICRA.2011.5980223.

Cumhur Erkan Tuncali and Georgios Fainekos. Rapidly-exploring Random Trees for Testing Automated Vehicles. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 661–666, 2019. doi:10.1109/ITSC.2019.8917375.

Unmesh Patil, Alessandro Renzaglia, Anshul Paigwar, and Christian Laugier. Real-time collision risk estimation based on stochastic reachability spaces. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 216–221, 2021. doi:10.1109/ICAR53236.2021.9659485.

M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477 Vol. 2, 1999. doi:10.1109/CEC.1999.782657.

Qiang Qiu, Man Yuan, Fei He, and Jinyun Fang. A parallel algorithm for line segment intersection. In *2013 21st International Conference on Geoinformatics*, pages 1–4, 2013. doi:10.1109/Geoinformatics.2013.6626128.

Yu Jiang, Yin-tian Liu, and Fan Zhang. An efficient algorithm for constructing Delaunay triangulation. In *2010 2nd IEEE International Conference on Information Management and Engineering*, pages 600–63, 2010. doi:10.1109/ICIME.2010.5478228.