LiDAR-based Cone Extraction and Heuristic Trajectory Optimization for Autonomous Formula Student Racing

Marko Jurić, Ante Vinković, Vlaho-Josip Štironja, Ivan Marković
University of Zagreb Faculty of Electrical Engineering and Computing
Laboratory for Autonomous Systems and Mobile Robotics (LAMOR)
Unska 3, HR-10000, Zagreb, Croatia
markojura11@gmail.com, ante.vinkovic99@gmail.com, vlaho-josip.stironja@fer.unizg.hr, ivan.markovic@fer.unizg.hr

***Abstract*—Addressing the challenges of autonomous racing in Formula Student competitions, this work proposes a novel LiDAR-based perception and trajectory optimization pipeline. Our key contributions include a robust cone detection algorithm for accurate track-edge identification from point clouds. These detections are integrated into a global map using an adapted FASTSLAM 2.0 framework, enhancing mapping accuracy in dynamic environments. Furthermore, we introduce a heuristic trajectory planner that minimizes path length and curvature trade-offs via cubic splines and a BFGS-based solver, and crucially, an offline regression model for real-time optimal weight estimation. The complete system is validated on real-world tracks with RTK-based ground truth, demonstrating superior mapping precision and efficient trajectory generation for high-performance autonomous navigation.**

***Keywords*—LiDAR; Cone Detection; Trajectory Optimization; Autonomous Racing; Formula Student**

I. INTRODUCTION

Autonomous racing in constrained environments presents unique challenges in perception, mapping, and planning. This paper presents a LiDAR-based pipeline that integrates cone extraction, Simultaneous Localization and Mapping (SLAM), and heuristic trajectory optimization to enable real-time navigation on complex race tracks. The approach is designed as part of a larger autonomous system pipeline that applies multiple sensor fusion techniques to maximize sensor data utilization and enhance overall system performance. This research was developed for the Formula Student Driverless (FSD) competition, an engineering competition focused on designing and building open-wheel race cars. The autonomous pipeline was implemented and tested on RT06-VulpesD, a fully autonomous electric race car developed by the FSB Racing Team at the University of Zagreb in compliance with the FSD rulebook [1].

In FSD context, to generate an accurate feature map SLAM algorithms usually process pre-filtered LiDAR observations. Thus, inspired by the work of Kabzan et al. [2] and initially outlined in [3], an algorithm for extracting points from a point cloud belonging to racetrack edge-marking traffic cones has been designed and expanded upon in the present paper. Thereafter, following the cone extraction step, a global map of cones is constructed using an implementation of the FASTSLAM 2.0 algorithm, in which the features making up the global map are exclusively detected traffic cones. This global map then serves as an input to the trajectory optimization algorithm proposed in this paper.

Trajectory optimization follows a heuristic approach to minimize a cost function, avoiding direct lap-time optimization due to its computational cost on long tracks with many cones. The cost function combines racing line heuristics, constrained by track edges. The resulting trajectory is modelled as a cubic spline to ensure smooth curvature transitions [4]. Once formed, the path is discretized at a user-defined interval, where finer discretization improves accuracy but increases computational complexity [4]. A key challenge of the heuristic approach is to ensure that its solution closely approximates the result of direct lap-time minimization, while preserving low computational complexity suitable for real-time use from the second lap onward [5]. After defining the path and optimization method, velocity profiling, based on the work of Velenis and Tsiotras [6], assigns velocities to each path segment. The final trajectory must also respect vehicle constraints, modelled using a dynamic two-track model that accounts for vertical load transfer and tire forces, as well as engine torque limitations.

Finally, cone extraction for SLAM was tested in Robot Operating System (ROS) on a real-world track adhering to FSD event track standards. The ground truth cone positions are extracted and transformed from the global to the local coordinate system in order to test the trajectory optimization. Ground truth cone data is collected using the Real-time kinematic positioning (RTK), reducing the average error to 15 cm. This example serves to showcase the performance of the proposed trajectory optimization method on a real-world use-case.

This research has been supported by the European Regional Development Fund under the grant PK.1.1.10.0007 (DATACROSS).

979-8-3315-9679-8/25/$31.00 ©2025 IEEE

II. Problem formulation and implementation

The pipeline processes perception and odometry data through FASTSLAM 2.0. Upon loop closure, the resulting cone map is used for trajectory optimization, yielding the trajectory, optimal slip angles, and yaw rates for each path segment, which serve as references for the Model Predictive Control (MPC).

A. Cone extraction

The cone extraction pipeline processes the input point cloud through several stages. First, the cloud is rotated to align with the sensor's position on the Autonomous Ground Vehicle (AGV). Passthrough filters then define a 3-dimensional subspace to reduce point density. Ground removal follows, using RANSAC to fit a dominant planar surface—assumed to be the ground—based on prior filtering and sensor geometry. Inlier points form the ground set, while outliers are passed to object detection. Ground removal is essential for enabling robust clustering in subsequent stages, as it prevents ground points from erroneously merging with above-ground objects in Euclidean distance-based clustering (DBSCAN). To recover cone base points mistakenly removed as ground, especially at longer distances where cones intersect few LiDAR beams, a cylindrical restoration step is applied. Sparse vertical cone structures often lose lower points during ground removal. To address this, DBSCAN first clusters the remaining cone candidate points. Then, cylindrical volumes (matching cone dimensions) are placed at cluster centroids, and ground inliers within these cylinders, queried via a KDTree, are re-added to the candidate set. The updated clusters are then validated to confirm they represent cones.

The validation framework uses a series of binary classifiers to assess whether each cluster represents a cone. Clusters are rejected if they fail any test. Validation includes analyzing cluster covariance across dimensions, applying a centroid height check $c_z \leq h_{threshold}$ (where c_z represents the centorid's position in the z-axis) to filter out high false positives, and using a second-degree polynomial to relate cluster distance to vertical spread. A final check evaluates cluster dimensions via a 3D bounding box, using the diagonal length as a plausibility metric for cone shape.

Figure 1 shows the cone extraction output from a single frame. Despite using a 16-channel LiDAR, reliable cone detection is achieved up to 20 meters, providing sufficient input for map construction in the AGV pipeline.

B. Cone extraction role in mapping

FASTSLAM 2.0 runs in three steps per iteration [7]: extending the path posterior, updating landmark estimates, and resampling. Path extension uses odometry from fused Inertial Measurement Unit (IMU) and wheel encoder data, adding Gaussian noise to each particle's x, y, and heading to preserve linear complexity and support data association. In the update step, cone detections act as landmarks, refining each particle's map through nearest-neighbor matching. Accurate cone extraction is essential, as false positives distort maps, noise affects associations, and false negatives increase landmark uncertainty. Resampling then favors particles with higher observation consistency, helping to mitigate errors and propagate the most likely hypotheses.

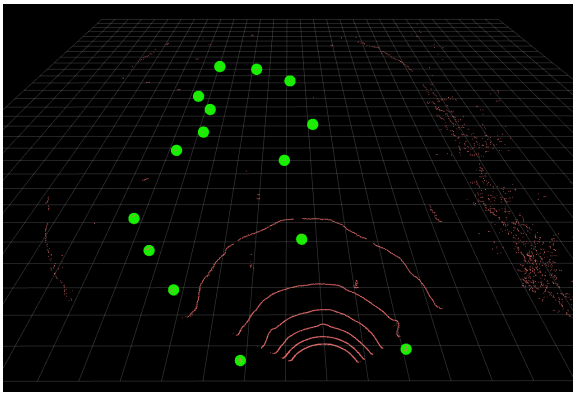


Figure 1. Cone extraction pipeline visualisation with input point cloud shown in red and detected cone centroids in green.

C. Trajectory optimization

The model is based on lateral and longitudinal load transfer, where the vertical force is estimated at each discretized point of the circuit based on the lateral and longitudinal acceleration. Given F_z, the maximum theoretical longitudinal F_x or lateral tire force F_y can be computed using a tire dynamics model. Specifically, the Pacejka Magic Formula is used [8]:

$$F = D \sin\left[C \arctan\left\{B\alpha - E\left(B\alpha - \arctan(B\alpha)\right)\right\}\right] \quad (1)$$

where B, C, D and E represent fitting constants and F is a force or moment resulting from a slip angle α. To make vehicle model simpler, there is an assumption made that the vehicle is always driven on the optimal slip ratio for the given vertical force, giving the vehicle maximum longitudinal force available from the tire. Given the steady state condition of slip ratio the vehicle model is put into the equilibrium state for it. The lateral slip angle is defined based on the ratio of the tire's lateral and longitudinal velocity components.

The simulations of the tire *Hoosier 18"* longitudinal and lateral friction coefficients μ_x and μ_y with respect to the vertical forces on the tire are estimated with the optimal slip assumption. Next, when lateral and longitudinal velocities are available, lateral and longitudinal force transfer can be calculated, which gives the vertical forces acting on each of the tires. Vertical forces are calculated in each step to determine the maximum traction the vehicle can produce at that control point. The maximum traction is divided into both lateral and longitudinal orientations and presented as maximum lateral force and longitudinal force of the tire ($F_{y,\max}$ and $F_{x,\max}$).

$$F_{x,\max,i} = mu_x \cdot x \cdot F_{z_i} \quad (2)$$

$$F_{y,\max,i} = mu_y \cdot x \cdot F_{z_i}, \quad (3)$$

where x denotes the friction utilization ratio, assumed to be approximately $\frac{2}{3}$ for dry asphalt.

Besides the friction constraint, the engine configuration and capability need to be modelled. The longitudinal acceleration

is limited by the engine torque. It can be derived from the torque curve – a function of engine torque in relation to engine speed.

To calculate maximum power the engine can output for the target velocity, firstly, engine velocity v_{engine} needs to be calculated using the wheel velocity v_{wheel} and transmission ratio (R) between its speed and wheel velocity:

$$v_{wheel} = \frac{v_{target} \cdot 60}{2\pi r_{wheel}}, \quad v_{engine} = R \cdot v_{wheel}. \tag{4}$$

Now, the longitudinal force can be derived without resistance. Corresponding engine torque is interpolated from the engine torque curve, which leads to the equation:

$$F_{x,e} = \frac{T_e \cdot R}{r_{wheel}}. \tag{5}$$

Finally, to achieve the most-accurate estimation of the longitudinal force vehicle can transfer to the ground, resistances are accounted for. Air and rolling resistance, modelled by standard aerodynamic and rolling force equations, are the primary resistive forces considered, and are subtracted from the (5) to compute the final longitudinal force at the given control point, while numerous other minor resistances are neglected.

Velocity concatenation to the given path is done using the three-pass filter. The theoretical maximum velocity is calculated using the engine speed (rpm) at the maximum torque value from the torque-engine speed curve. Final equations are:

$$v_w = \frac{v_{e_{max}}}{R}, \quad v_{x_{max}} = \frac{v_w \cdot 2 \cdot \pi \cdot r_w}{60}. \tag{6}$$

The maximum theoretical longitudinal velocity is then fed into the first-pass filter.

$$F_{lat} = F_{friction} \rightarrow \frac{mv^2}{r} = \mu_y mg \rightarrow v = \sqrt{r\mu_y g}, \tag{7}$$

local velocity can be derived, and it depends solely on the lateral friction coefficient and radius. To find the adequate μ_y, the minimum one is used from the load transfer.

The second filter applies forward integration, updating velocities under positive acceleration by considering engine, traction, and lateral constraints. The minimum of these limits proceeds to the final filter:

$$v_{acclim} = min(v_{local}, \sqrt{v_0^2 + 2 \cdot a_x \Delta s}). \tag{8}$$

The third pass performs backward integration to ensure safe deceleration using only longitudinal friction. For unevenly paired cones, the midline is estimated by connecting the nearest cones of opposite color and computing their center. A normalized parameter $\alpha \in [0, 1]$ indicates the intersection point along this segment. To ensure safety, the lateral track width is reduced by the cone radius, half the vehicle width, and the expected localization error.

To generate a smooth trajectory, concatenated cubic splines are used. However, as noted in [9], these splines do not guarantee global convexity along their entire length. As a result, standard gradient descent methods may converge to suboptimal local minima. To address this, the BFGS (Broyden–Fletcher–Goldfarb–Shanno) algorithm is employed as the trajectory optimization method.

The trajectory optimization problem is formulated to minimize a weighted combination of two objectives: total path length and average path curvature. The scalar cost function is defined as:

$$\begin{aligned} \text{Minimize} \quad & F(\varepsilon) = (1-\varepsilon) \cdot \Gamma^2 + \varepsilon \cdot S \\ \text{Subject to} \quad & t(s), \end{aligned}$$

where ε denotes weight factor, Γ average path curvature and S path length. Now, that parameter will be the centre of optimization problem and the subproblem will be to determine the α values mentioned above. Under the hood, ϵ factor will be optimized based on lap time. To further reduce online computational effort, an offline regression-based estimation of the optimal ε is introduced. Empirical analysis across a diverse dataset of test tracks showed that the best predictor of the optimal weight was the mean curvature through corners. A linear regression model is trained using this feature, resulting in a predictive model for ε that generalizes well across various track types. The dataset used for training includes hand-drawn layouts, randomly generated tracks, and real-world tracks reconstructed from Trackdrive event data. To prevent overfitting on limited data, a linear regression model was fitted using *sklearn* and used as input to the direct optimization method.

D. Collecting the ground truth track data

Ground truth data is essential for evaluating the pipeline on a real-world track by comparing SLAM outputs to true cone positions after each lap. Evaluation uses confusion matrices and RMSE of correctly matched cones. Cone positions were recorded using an RTK GNSS module, with the receiver placed at each cone base for approximately 5 s, collecting dense, high-precision data at a frequency of approximately 10 .

DBSCAN clustering is applied to the RTK-collected point set in the global coordinate system, with DBSCAN parameters ϵ and n determined by cone size and RTK frequency. The mean of each resulting cluster provides a accurate ground truth cone location. To evaluate SLAM and map accuracy, ground truth data is transformed from global to local coordinates, using the midpoint between the starting orange cones as the origin. After transformation, the cones are manually classified into blue and yellow sets based on their physical layout, and any missing cones are annotated or reconstructed as needed to ensure alignment with the test setup. Process is depicted within the Fig 2.

III. RESULTS

Specifically, all perception software is developed for a 16-channel Velodyne Puck Hi-res LiDAR sensor. In addition to that, the entire pipeline uses the IMU, wheel encoders, and RTK module for precise GNSS. The IMU used in the race car is an Xsens MTi-680G RTK GNSS/INS. The cone extraction

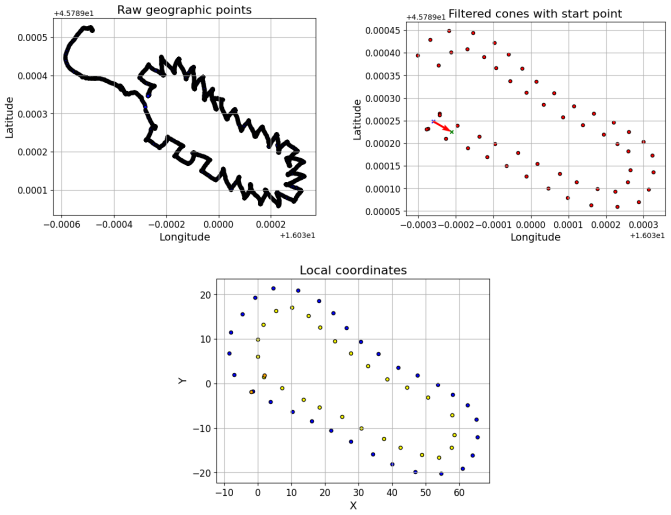


Figure 2. Postprocessing of the ground truth track data from raw positions, cones extracted in GNSS frame and in the local frame filled with missing ones.

algorithm is mostly visually validated while looking at each timestamp. In the each timestamp's raw pointcloud it is clear which clusters are representing the cones and has the algorithm correctly classified it. Besides that, the feasible cone estimation range can be easily calculated due to Velodyne's LiDAR having 16 rays with vertical resolution of 1.33°. Knowing the height of the cone and vertical resolution there is a fixed distance where certainly will not be two rays intersecting the cone:

$$d = \frac{360 \cdot h_c}{2\pi \cdot \alpha}, \tag{9}$$

where h_c is the cone height and the α is the vertical resolution. When the numbers are put in the equation, the maximum distance the LiDAR has two rays intersecting the cone is 12.92 m. That way, while visually validating the cone extraction, all cones outside that scope can be disregarded and excluded from pointcloud. Furthermore, cone extraction should be validated within the SLAM performance, even though its performance does not solely depends on it. Currently, the SLAM validation metrics have only been calculated on the FSSIM (*Formula Student Simulator*) and not on the real track data. RMSE on the FSD track was 0.35 m for true positive classifications. Matrices are shown in Table I. The RMSE measurement provides a

TABLE I. Confusion Matrices for Blue and Yellow Cones

	Blue		Yellow	
Actual \ Predicted	**Positive**	**Negative**	**Positive**	**Negative**
Positive	87	5	86	10
Negative	0	X	2	X

useful basis for evaluating how cone localization accuracy affects trajectory planning performance. Furthermore, lap times were computed under two scenarios: one assuming perfect cone localization, and the other incorporating the observed RMSE (0.35 m) by artificially expanding the track boundaries to account for positional uncertainty.

Two objective function strategies, as described in Section II, were evaluated: an online compromise method and an offline variant. While the offline method results in a slight performance penalty, yielding an average lap time increase of only 0.32% (approximately 0.1 seconds on a 30-second lap), it offers a significant computational advantage. Specifically, the offline method requires only 4 seconds to compute, compared to approximately 80 seconds for the online approach. Under ideal conditions, the lap times were 10.46 s (online) and 10.49 s (offline), while with RMSE included, the times were 10.62 s and 10.66 s, respectively.

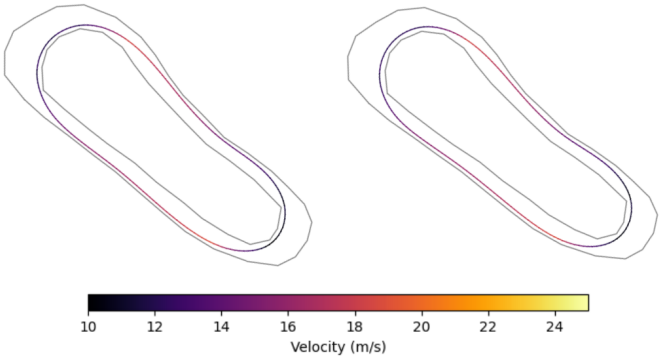


Figure 3. Comparison of optimal trajectory between perfect cone extraction and 0.35 RMSE estimation.

IV. Conclusion

This work presented a LiDAR-based cone detection and SLAM framework integrated with a heuristic trajectory optimization pipeline for autonomous Formula Student racing. The proposed method enables real-time performance by combining FASTSLAM 2.0 with an offline-learned cost weighting model. The trajectory optimization algorithm fixed the long computational time direct optimization had with the clever dataset aggregation while compromising the 0.32% of lap-time on average. Future work includes extensive data collection regarding track tests and vehicle model improvement, and on-vehicle deployment to further evaluate and improve the algorithm's performance and generalization capabilities.

References

[1] Formula Student Rules 2025, Version: 1.1, Formula Student Germany, 11 2024.
[2] Kabzan et al., "AMZ Driverless: The full autonomous racing system" Journal of Field Robotics, vol. 37, 10.1002/rob.21977, 08 2020
[3] Vinković, Ante, "Accurate traffic cone detection in racing environments based on a LIDAR sensor", Bachelor's thesis, 2022
[4] G. J. C. Kapania, Nitin R; Subosits, "A sequential two-step algorithm for fast generation of vehicle racing trajectories," Cornell University Library, 2019.
[5] Alexander Heilmeier and Maximilian Geisslinger and Johannes Betz, A Quasi-Steady-State Lap Time Simulation for Electrified Race Cars, IEEE, 2019.
[6] Velenis and P. Tsiotras, "Minimum time vs maximum exit velocity path optimization during cornering," Georgia Institute of Technology, 2005.
[7] S. Thrun, "Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data association," Computer Science Department Stanford University, 2003.
[8] Hans B. Pacejka, Chapter 9 - Short Wavelength Intermediate Frequency Tire Model, Editor(s): Hans B. Pacejka, Tire and Vehicle Dynamics (Third Edition), Butterworth-Heinemann, 2012, Pages 403-474.
[9] J. R. Chasnov, Numerical methods. Hong Kong University of Science and Technology, 2024, Pages 30-33