Reactive control algorithm for F1TENTH Autonomous Vehicles in unknown dynamic environments

Artur Morys - Magiera[1*], Aleksandra Lis[1], Jakub Pudło[1], Andrzej Papierok[1], Marek Długosz[1*] and Paweł Skruch[1]

[1*]Department of Automatic Control and Robotics, Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering at AGH University of Science and Technology, Mickiewicza 30 av., Kraków, 30-059, Lesser Poland, Poland.

*Corresponding author(s). E-mail(s): amorys@student.agh.edu.pl; mdlugosz@agh.edu.pl;
Contributing authors: alis@student.agh.edu.pl; jpudlo@student.agh.edu.pl;
andrzejp@student.agh.edu.pl; pawel.skruch@agh.edu.pl;

Abstract

Autonomous control of scaled racing cars is an increasingly popular approach for testing and presenting new control algorithms while also reducing the operational costs required to bring such a model along with a proper racing track to life. In most cases where autonomous vehicles come into play, the environment map is not known a priori, in which case the algorithm should sufficiently drive the car in real time, without having planned an optimal trajectory. This study presents a reactive method based solely on LiDAR readings that successfully navigates the car through unknown dynamic environments with obstacles, taking into account the principle of safety: not to crash into obstacles or other vehicles. We show that the algorithm significantly outperforms two other existing reactive algorithms in terms of lap time and in terms of driving trajectory length. The proposed algorithm renders a cost-effective solution to the problem of autonomous driving on unknown maps, as well as it renders itself as a possibly good "specialist" model in Expert Intervention Learning (EIL) approach-based models for autonomous vehicles, especially the F1TENTH cars.

Keywords: autonomous, neural network, race

1 Introduction

Motorsport racing requires the driver to be able to react quickly to the track and the dynamic changes that may occur during the race. Thus, all decisions have to be made and executed in fractions of a second. However, in case of a dynamic environment, the map may have altered since the start of the race, may become cluttered with obstacles or other vehicles, or may be unknown a priori until the beginning of the race. In addition, vehicles, especially those designed to be smaller in scale, such as F1TENTH, have a limited amount of computing resources. These constraints may render existing navigation algorithms that are based on driving on a track mapped beforehand, then localizing the robot and planning a trajectory for it to follow less suitable or even inapplicable.

The goal of this article is to propose a control algorithm capable of navigating the car with safety mechanisms allowing for dodging obstacles and other vehicles and recuperating in case of landing

in an orientation from where it cannot continue driving forward. Additionally, we assume that any racing track for this algorithm to drive on is in accordance with the F1TENTH rules [1] and has no dead ends; however, it may be of non-uniform width, for example, it may have wavy walls. We finally assume - which is crucial to the topic of this article - that the driving track progression is unknown, which in turn implies an online control approach in contrast to an offline-generated trajectory to be followed over a mapped track.

The article is organized as follows:

(I) Section 2 - "F1TENTH autonomous racing" provides an insight into the genesis, scope, and goal of the F1TENTH formula, which provides the core goals, assumptions, and limitations that our autonomous robot follows. It also describes the technical details of our F1TENTH setup and presents a real world verification of our F1TENTH car build controlled by the algorithm proposed in this article during the '22 KSMTE's (The Korean Society of Manufacturing Technology Engineers) The 1st Korea International F1TENTH Championship [7], where we scored the 3rd award.

(II) Section 3 - "Related work" describes similar other research related to the topic of this article.

(III) Section 4 - "Operating principle" presents the key concepts of the control algorithm, divided into the following subsections.

(a) Section 4.1 - "LiDAR data acquisition, preprocessing and real-world verification" gives an idea of the characteristics of the LiDAR data on which the algorithm operates and how it is processed.

(b) Section 4.2 - "Localization of danger zones" desribes the process of localizing what we refer to as "danger zones", which are areas around the car that driving in is not viable, as it would result in crashing into an obstacle while turning nearby.

(c) Section 4.3 - "Safety zones around the vehicle" concerns utilizing LiDAR readings within areas we suggest as "safety zones" with priority higher than the primary navigation algorithm to be used for navigation, avoiding crashes, and intervention in edge cases.

(d) Section 4.4 - "Navigation strategies" presents the flow of the four navigator variants, in their respective priority order (highest-to-lowest); if a higher-priority navigator variant is not applicable in a given context, then the next one with lower priority is used.

(i) Section 4.4.1 - "Front collision / recuperation" describes the navigator handling recovery from no-drive situations (i.e., when the car is stuck in front of the wall and there is too little space to make a single turn) and handles cases when an obstacle in the front is extremely close.

(ii) Section 4.4.2 - "Safety zone violation" describes the navigator handling a presence of an obstacle (violation) in a safety zone around the vehicle for the car to react appropriately.

(iii) Section 4.4.3 - "Sharp turning" presents a navigator that seizes control when the car is located nearby a sharp turn and may drive more aggressively to optimize the path distance and time.

(iv) Section 4.4.4 - "Casual driving" refers to the primary navigator that is the core of the proposed algorithm and takes over control when neither of the former variants becomes active.

(e) Section 4.5 - "Determination of driving speed" describes the way driving speed is calculated based on the calculated steering and the environment context.

(IV) Section 5 - "Implementation performance" describes the performance of our implementation along with the characteristics of the data sources from the environment.

(V) Section 6 - "Comparison with existing solutions" compares the algorithm with two existing classical control algorithms in a simulated racing environment.

(VI) Section 7 - "Conclusion" sums up the conclusions coming from the information presented in this article and provides insight into the future research paths that our contribution opens in the field of reactive algorithms applied in autonomous racing in unknown environments.

2 F1TENTH autonomous racing

F1tenth is an international formula race competition for autonomous vehicles initiated by researchers at the University of Pennsylvania [4, 10]. The project gathers more than 60 universities worldwide as well as research institutions that construct their own vehicles based on the common rules established by this international community. The aim of the community is to develop new algorithms with the aim to later compete in races.

An F1TENTH racing competition itself is divided into 3 parts, in compliance with the general established rules:

1. Test session: each team is given a certain amount of time by the organizers to get accustomed to the track and make minor adjustments to their algorithms.
2. Time trials: in this part, the teams compete for the best time. Each team gets 2x5min for the ride. In this part of the competition, teams can score points in two ways: for the most laps completed within 5 minutes and for the fastest time for one lap. The number of points scored in this part determines the placing in the "HEAD-TO-HEAD" part.
3. Head-to-head races: in this part two teams race simultaneously on the track, the cars are placed on opposite sides of the track and the winner is the car which catches up with and overtakes the opponent or the one which completes the 10 laps faster.

For our F1TENTH build, we used a TRAXXAS Ford Fiesta 1/10 chassis. NVIDIA Jetson TX2 served as the main onboard computer, and we used the Hokyuo UST-10LX LiDAR sensor, as directed by the organizers' rules regarding the allowed hardware. The LiDAR sensor provides readings at a frequency of 40Hz with an angular resolution of 0.25°, a $\pm 40mm$ accuracy, and a scanning range of 270°. The setup is presented in fig. 1.

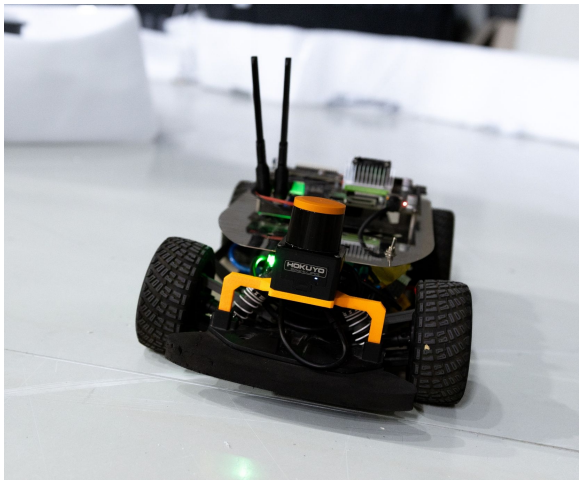


Fig. 1: Our F1TENTH vehicle setup; photography courtesy of Mateusz Jurczak

The F1TENTH setup also covers design and assembly of a dedicated Power Distribution Board (PDB) to provide energy to all onboard electronic components. The PDB connects the VESC motor driver to the battery and converts the voltage to appropriate levels for powering other components using step-up and step-down converters. The NVIDIA Jetson is powered with 18V, the LiDAR is supplied with 12V and auxiliary 5V outputs is also provided for the USB HUB external power source. The board also features informational LEDs and fuses for circuit safety. Figure 2 presents the design of the PDB used in our build, while fig. 3 presents the real assembled board.

It is noteworthy that in the '22 KSMTE's The 1st Korea International F1TENTH Championship [7] our team scored the 3rd award, driven by a reactive algorithm not taking into account the test session, competing with the builds that utilized mapping and localization techniques.

3 Related work

The F1TENTH society has put forward propositions of autonomous driving algorithms.

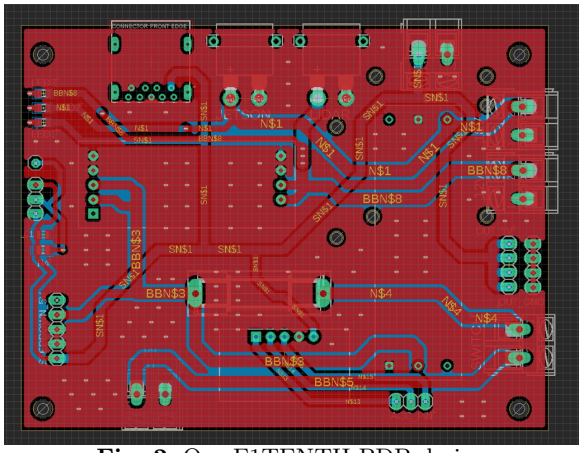


Fig. 2: Our F1TENTH PDB design

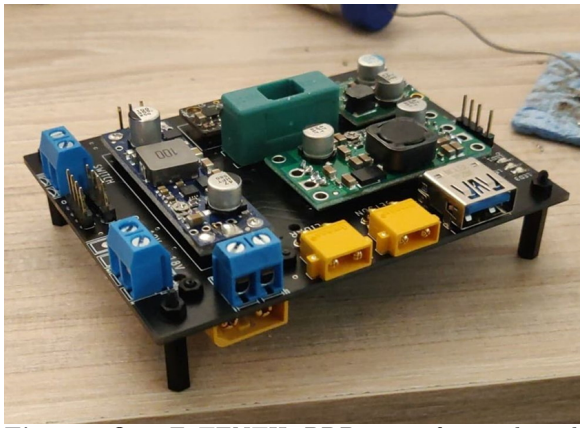


Fig. 3: Our F1TENTH PDB manufactured and assembled

To the simplest algorithms that require no previous configuration belongs Follow The Gap (FTG) [13], a reactive algorithm requiring little configuration (safety bubble threshold, horizon length, steering reactivity coefficient and speed control configuration) that follows the greatest (widest or furthest) gap in the LiDAR readings. Another example is Follow The Wall (FTW) algorithm [15], which follows the wall and also requires little configuration (PID controller parameters, wall follow distances from both sides, horizon length and speed control configuration).

One methodology that is steadily improving thanks to the rise of popularity of deep learning and Deep Neural Networks (DNNs) is end-to-end driving - a concept in which all data from sensors are passed to a DNN which in turn outputs the driving parameters directly - in terms of F1TENTH: the steering angle ϕ and velocity v, such as in [9].

A variation of this method - partial end-to-end driving - involves combining the DNN with other modules in a cascade model, where the outputs of the DNN flow into the other modules, which in turn pass the data on, up to the actuators. Such attempts were carried out in [17, 18].

Another approach is Model-Predictive Control (MPC), which makes use of dynamic models of vehicles to perform actuation on the vehicle [2]. One research proposes a combination of MPC with Convolutional Neural Networks (CNNs) to perform an attention-based predictive control [8].

4 Operating principle

The algorithm consists of 7 stages connected in a pipeline as shown in fig. 4.

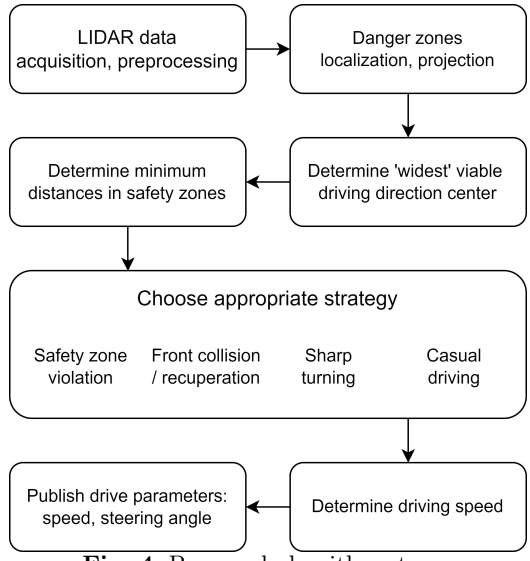


Fig. 4: Proposed algorithm stages

4.1 LiDAR data acquisition, preprocessing and real-world verification

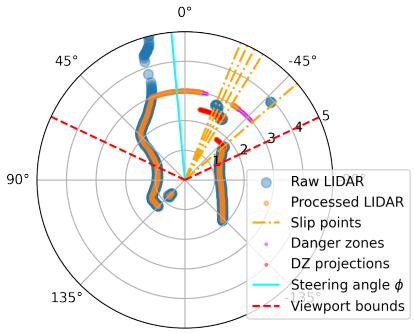
LiDAR data often come with missing data - which require imputation - and noisy data - which require proper preprocessing - before passing them to the navigation algorithms. Other pieces of research also focus on data imputation and processing using methods ranging from classical

statistics [19] through linear regression [16], K-Nearest Neighbours (KNN) regression [16, 19] and finally reaching full algorithms preprocessing the data and using the data to perform autonomous control [5, 6].

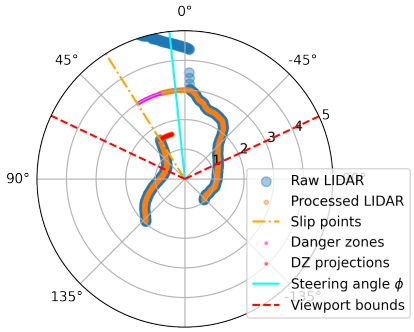
Our approach limits all distances to a set horizon H by clipping all values (beam lengths) that exceed this threshold. This operation makes all points fit inside the horizon, eliminating discrepancies at larger ranges. Furthermore, beams hitting obstacles that are too close or too far carry `NaN` values, which we suggest replacing with the maximum values. Finally, an important feature of the data is that beams for subsequent points at larger distances naturally correspond to points that are spread further than points at smaller distances, which can be seen in fig. 5a. That same figure also presents noise: malformed laser readings (northeast).

Another two noteworthy situations include fig. 5b, where some of the points are visible far away above the horizon, as well as some points are invisible due to the almost-vertical curvature of the wall on the north, and fig. 5c, where the data are generally clear (contains insignificant noise data on the south), but the valid track involves a "dead sinus" that might cause the car to unnecessarily turn more to the left - which is remediated with the danger zones mechanism described in section 4.2.

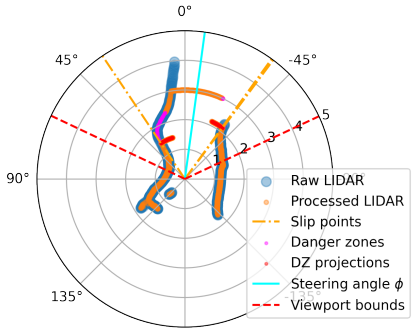
The plots in fig. 5 present real-world data collected during the International 1st F1TENTH Korea Championship [7], in which our team scored the 3rd award with the car driven by the algorithm described in this article. The plots were prepared with the horizon length H set to $3m$.



(a) Twisted track (noisy sample with open space)



(b) Close left turn (danger of hitting the inner wall with the left side while turning)



(c) "Dead" sinus (located northwest), which can trick the car into driving wobbly

Fig. 5: Real-world track frames from the '22 KSMTE's The 1st Korea International F1TENTH Championship processed by the proposed algorithm

4.2 Localization of danger zones

We search for danger zones, which are zones where driving would be risky, although the trajectory might be chosen as optimal by the navigation algorithm. Examples include situations in fig. 5b,

where the car would have hit the left track boundary with its side if it took a sharp turn, or in fig. 5c, where the car might have lost time by going into the direction of the "dead sinus" present on the track. With the dead zones mechanism engaged, both of these problems are remediated. The mechanism is similar to what other researchers implemented in their variant of the proposal of the F1TENTH vehicle control algorithm [11]. Within the subsequent points in a reading of the LiDAR, samples that differ drastically in terms of their distance from the sensor are found. Then, these slips' indices are saved along with the distance of the closer of the points in a pair. The polar sector between the arcs at the distance of the closer and further points becomes a danger zone. The subfigures in fig. 5 depict the danger zones and their projections, which are the areas that they cover at further distance.

The exact process for determining the length of a danger zone is based on the size of the car and a safety margin. However, the polar coordinates plane that is natural for processing LiDAR readings imposes the need to take into account perspective: the car projection at closer distances is of different span (in the sense of the number of beams that are covered) than at further distances. Therefore, the coordinates of a slip between subsequent points are first converted from polar to Cartesian coordinates (where the problem of perspective no longer persists), then are translated on the y-axis by the car size and a safety margin, and finally are converted back from Cartesian to polar for further processing. This procedure compensates for the perspective in the polar coordinate system.

4.3 Safety zones around the vehicle

For both speed control as well as emergency braking and turning in case dynamic obstacles appear unexpectedly close, we propose to divide the area around the car into sub-areas as shown in fig. 6.

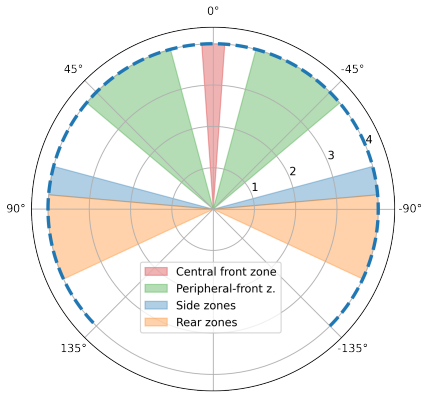


Fig. 6: Proposed safety zones around the car

For each of the safety zones, the minimum range is compared with a safety threshold assigned to the given zone, and the results (referred to as "alarms") have an impact on the navigation strategy used for driving, as described in the next section.

4.4 Navigation strategies

Based on the presence of alarms in the safety zones, the minimal distances, and the validity of readings, an optimal navigation strategy is chosen.

4.4.1 Front collision / recuperation

If the alarm in the central front zone is active, the car will turn aggressively to the side, where the maximum beam range in the central front zone (red area in fig. 6) is greater - this is a guess of a better choice for driving. However, if the minimum value in this zone is lower than a minimum threshold, below which there is no space for the car to turn without colliding, the car jumps backwards to recover from this standoff. The algorithm for this variant is presented in fig. 7.

4.4.2 Safety zone violation

If an obstacle exceeds a minimum distance threshold in any of the side or rear zones (respectively: orange and green areas on fig. 6), the car aggressively turns in the opposite direction; if alarms from both sides are exceeded simultaneously, the car continues driving straight. The algorithm for this variant is presented in fig. 8.

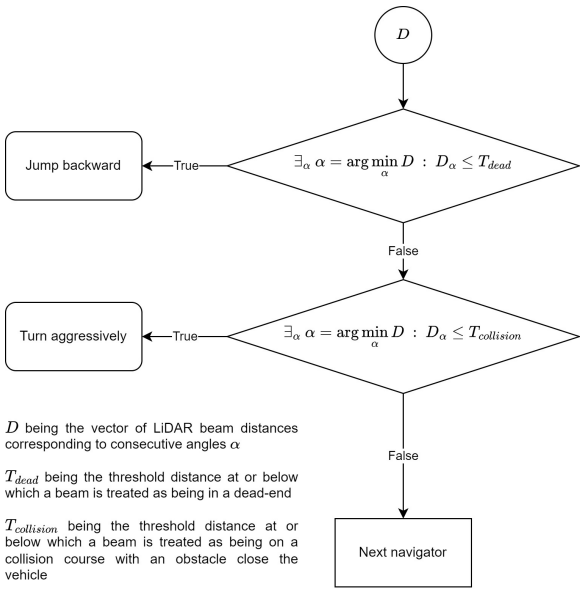


Fig. 7: Front collision / recuperation navigator algorithm

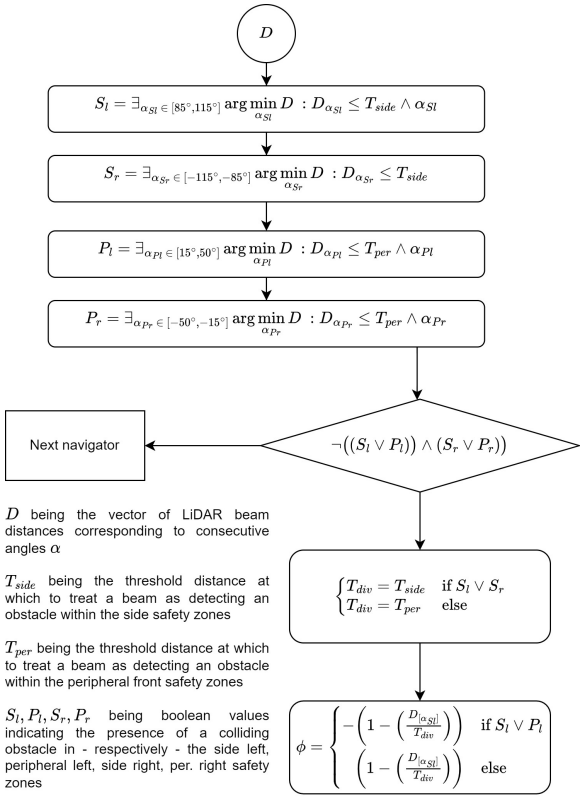


Fig. 8: Safety zone violation navigator algorithm

4.4.3 Sharp turning

If the distance in either (or specifically both) of the side zones (blue areas in fig. 6) is below a set threshold and also neither of the beams is out of range of the LiDAR, which means that the data should be reliable, the side with the higher maximum beam distance is chosen and a fixed sharp turn steering is applied. If data only from one of the sides is reliable and satisfies the threshold condition, this side is picked immediately. Otherwise, this navigator is inactive. The algorithm for this variant is presented in fig. 9.

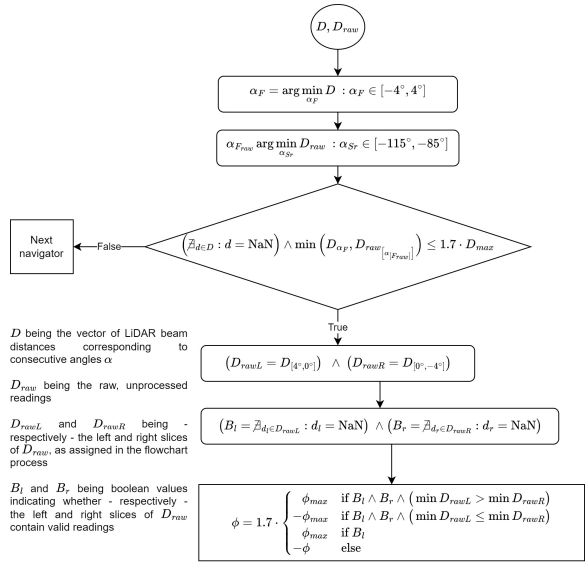


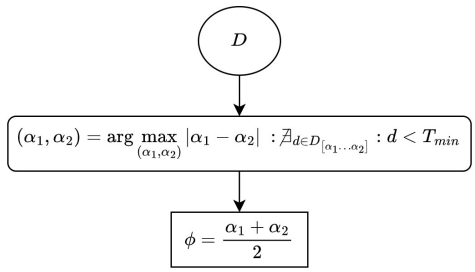
Fig. 9: Sharp turning navigator algorithm

4.4.4 Casual driving

If neither of the navigators above is active, the casual (primary) navigator is active. This navigator picks the longest sequence (in the sense of consecutive laser beam samples) of samples with maximum distance d_{max}, constrained to satisfy a minimum threshold. The steering angle is then set in the middle of this sequence. The algorithm for this variant is presented in fig. 10.

4.5 Determination of driving speed

We propose a non-linear relation between a normalized error metric e and the setpoint of the car



D being the vector of LiDAR beam distances corresponding to consecutive angles α

T_{min} being the minimum distance threshold

Fig. 10: Primary navigator algorithm

speed v, as in eq. (1).

$$v(e) : e \in [0;\ \ 1] \wedge\ \ v \in [0;\ \ v_{max}] \tag{1}$$

For ease of configuration, we implemented the interpolation of $v(e)$ using the cubic spline, as its progression is smooth (its first two derivatives are continuous in its entire domain). As an error metric, we used a function of the navigation steering angle ϕ and, if present, the maximum beam distance of the primary navigation strategy.

$$\begin{aligned} &d_{max} : e(\phi,\ d_{max}) = \\ &= \mathbf{max}\Big(\frac{|\phi|}{\phi_{max}}, \big(1 - \mathbf{min}(1,\ d_{max} - d_{range})\big)\Big) \end{aligned} \tag{2}$$

where d_{range} is the maximum operating distance of the LiDAR. We propose the interpolation $v(e)$ to be a cubic spline presented in fig. 11, which has proved to be successful during the '22 Korea International F1TENTH Championship.

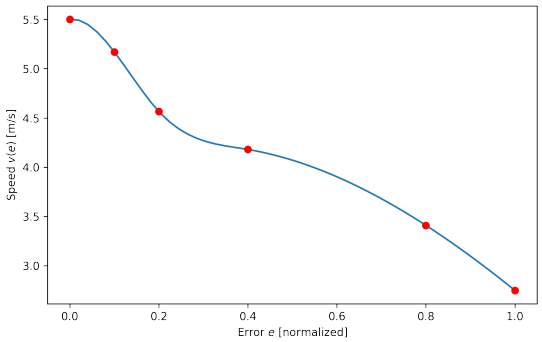


Fig. 11: Proposed speed interpolation cubic spline

5 Implementation performance

The algorithm has been implemented on top of the ROS (Robot Operating System) framework in Python as two modules: one being the core node, serving the navigation logic, and another one publishing the steering speed to the VESC motor controller. The reason for existence of a second node is the need for publishing control commands at a proper required, greater than the logic node, frequency. Since the core node depends solely on the LiDAR data, it's rate is essentially synchronised with the LiDAR. For our F1TENTH-compliant solution, we use the Hokyuo UST 10LX LiDAR, which passes readings at a frequency of $40Hz$, angular resolution of $0.25°$ and a scanning range of $270°$, which result in the LiDAR providing a vector of 1081 readings (beams). The Python implementation running on an NVIDIA Jetson TX2 SBC (Single-Board Computer) achieves above $30Hz$ running our algorithm; however, due to the inefficient (in terms of performance) handling of memory in numpy ndarrays we used for calculations, this value shall improve when using preallocated buffers for LiDAR data storage and processing with more performant lower-level, precompiled and optimised languages, e.g. C++, which would relax the computation time.

6 Comparison with existing solutions

We carried out a comparison of our algorithm with two reactive algorithms available on GitHub: FTW [15] and FTG [13, 14]. The comparison was carried out on the 2020 Berlin Grand Prix race track [3] using the F1TENTH ROS Gym [12].

Hyperparameters for FTG were fine-tuned as follows: desired left-side distance $l = 0.8$, horizon length set to $H = 3m$, velocities set to 3.5, 2.5 and 1.5 - respectively - for: high, medium and low error values; for FTW these were: horizon length set to $H = 3m$, bubble size set to 1.0, reactivity set to 0.5 and speed - by analogy to FTG - set to: 3.0, 2.0, 1.0. The speed interpolation and the parameters of our algorithm remained as is, except for the horizon length (because the map is larger), which was increased to $H = 6m$. The trajectories are presented in fig. 12.

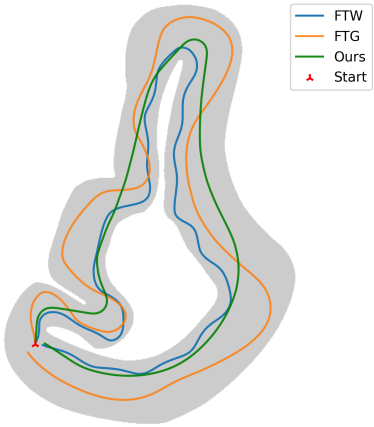


Fig. 12: Trajectories of the three algorithms on the "Berlin" map, logged from the F1TENTH ROS Gym simulation

Distances were calculated as the sum of Euclidean distances between consecutive points reported by the simulator, and times as the timestamp delta between the first and last frames logged. These metrics are shown in table 1.

Table 1: Control algorithms' metrics comparison

Algorithm	Lap length	Lap time
FTW	66.160m	23.384s
FTG	82.768m	26.345s
Our	**61.355m**	**12.728s**

7 Conclusion

The proposed algorithm stands as a reliable and safe method of autonomous driving in unknown, even dynamic environments and outperforms the classical reactive approaches (FTW, FTG). Finally, it provides a good starting point for further research on autonomous racing vehicle control with methods like Expert Intervention Learning, where we believe it can very easily be applied as the expert model to seize control when the trainee model performs badly, then recover the car and cease control back to the trainee. Furthermore, our algorithm would do so automatically, with very few configurations done beforehand (mainly the horizon length H value, increased proportionally to the general approximate size of the track). We plan to focus future work on RL (Reinforcement Learning) control models using the EIL methodology and the proposed reactive algorithm as the expert.

References

[1] Houssam Abbas, Madhur Behl, Matthew Brady, Phil Hu, Timothy Hu, Paril Jain, Paritosh Kelkar, Nischal K N, Rahul Mangharam, Liz W. P. Ng, Matthew O'Kelly, and Carter Sharer. F1/10 - the rules, version 1.0. URL https://f1tenth.org/misc-docs/rules.pdf.

[2] Vittorio Cataffo, Giuseppe Silano, Luigi Iannelli, Vicenç Puig, and Luigi Glielmo. A nonlinear model predictive control strategy for autonomous racing of scale vehicles. In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 100–105. IEEE, 2022.

[3] F1TENTH Contributors and Hongrui Zheng. Berlin grand prix map. URL https://github.com/f1tenth/f1tenth_simulator/blob/master/maps/berlin.png.

[4] F1TENTH Foundation. F1tenth. URL https://f1tenth.org/.

[5] Adam Gotlib, Kornelia Łukojć, and Mateusz Szczygielski. Localization-based software architecture for 1:10 scale autonomous car. In *2019 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 7–11, 2019. doi: 10.1109/IIPHDW.2019.8755418.

[6] Chan Wei Hsu, Tsung hua Hsu, and Kuang Jen Chang. Implementation of car-following system using lidar detection. *2012 12th International Conference on ITS Telecommunications*, pages 165–169, 2012.

[7] KSMTE. The 1st f1tenth korea championship. URL https://korea-race.f1tenth.org/index.html.

[8] Keuntaek Lee, Gabriel Nakajima An, Viacheslav Zakharov, and Evangelos A Theodorou. Perceptual attention-based predictive control. In *Conference on Robot Learning*, pages 220–232. PMLR, 2020.

[9] Yaqub Mahmoud, Yuichi Okuyama, Tomohide Fukuchi, Tanaka Kosuke, and Iori Ando. Optimizing deep-neural-network-driven autonomous race car using image scaling. In *SHS web of conferences*, volume 77, page 04002. EDP Sciences, 2020.

[10] Matthew O'Kelly, Varundev Sukhil, Houssam Abbas, Jack Harkins, Chris Kao, Yash Vardhan Pant, Rahul Mangharam, Dipshil Agarwal, Madhur Behl, Paolo Burgio, and Marko Bertogna. F1/10: An open-source autonomous cyber-physical platform, 2019. URL https://arxiv.org/abs/1901.08567.

[11] Nathan Otterness. The "disparity extender" algorithm, and f1/tenth. URL https://www.nathanotterness.com/2019/04/the-disparity-extender-algorithm-and.html.

[12] Matthew O'Kelly, Hongrui Zheng, Dhruv Karthik, and Rahul Mangharam. textscf1tenth: An open-source evaluation environment for continuous control and reinforcement learning. In *NeurIPS 2019 Competition and Demonstration Track*, pages 77–89. PMLR, 2020.

[13] Volkan Sezer and Metin Gokasan. A novel obstacle avoidance algorithm: "follow the gap method". *Robotics and Autonomous Systems*, 60(9):1123–1134, 2012. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2012.05.021. URL https://www.sciencedirect.com/science/article/pii/S0921889012000838.

[14] Yash Trikannad. f110_reactive_methods, . URL https://github.com/YashTrikannad/f110_reactive_methods.

[15] Yash Trikannad. f110_wall_follow, . URL https://github.com/YashTrikannad/f110_wall_follow.

[16] Christian Velasco-Gallego and Iraklis Lazakis. Real-time data-driven missing data imputation for short-term sensor data of marine systems. a comparative study. *Ocean Engineering*, 218:108261, 2020. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2020.108261. URL https://www.sciencedirect.com/science/article/pii/S0029801820311823.

[17] Trent Weiss and Madhur Behl. Deepracing: Parameterized trajectories for autonomous racing. *arXiv preprint arXiv:2005.05178*, 2020.

[18] Trent Weiss, John Chrosniak, and Madhur Behl. Towards multi-agent autonomous racing with the deepracing framework. In *International Conference on Robotics and Automation (ICRA)-Workshop on Opportunities and Challenges with Autonomous Racing*, 2021.

[19] Yifan Zhang and Peter J. Thorburn. Handling missing data in near real-time environmental monitoring: A system and a review of selected methods. *Future Generation Computer Systems*, 128:63–72, 2022. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2021.09.033. URL https://www.sciencedirect.com/science/article/pii/S0167739X21003794.