# Demo: Deadline-Aware Real-Time Planning on F1TENTH

Samir Gupta, Michael Yuhas, Qiaosi Lei, Abhishek Dubey
Vanderbilt University
Nashville, TN, USA

***Abstract*—We demonstrate a complete, real-time autonomous racing stack that deploys Deadline-Aware Monte Carlo Tree Search (DA-MCTS) on physical F1TENTH hardware powered by an NVIDIA Jetson Orin Nano. DA-MCTS adapts the risk profile of its sampled actions to the compute budget available at runtime, letting a sampling-based planner meet a hard control deadline on a resource-constrained processor without retuning. On F1TENTH hardware, DA-MCTS computes a racing line online and attains lap times competitive with pure-pursuit and Stanley baselines that require a precomputed reference line. Demo video link: https://youtu.be/vgiwadLAvKs and source code link: https://github.com/scope-lab-vu/RTCSA_Demo_ws.**

***Index Terms*—Real-time Path Planning, Monte Carlo Tree Search, Autonomous Racing, F1TENTH, Particle Filter Localization**

## I. Introduction

Autonomous racing presents a unique challenge, where actions must be chosen under nonlinear tire dynamics and noisy sensor data, within control deadlines that span only a few milliseconds. Iterative and sampling-based methods such as Model Predictive Control (MPC) and Monte Carlo Tree Search (MCTS) are traditionally adopted for this task, but their solution quality scales with the number of iterations completed per control step. This creates a problem at deployment, since such planners are tuned on workstations yet run on embedded processors (e.g., the NVIDIA Jetson Orin Nano), where the compute budget is smaller and varies at runtime as perception and localization compete for the same hardware. Deadline-Aware MCTS (DA-MCTS) addresses this by estimating the available compute budget online and shifting its continuous action space sampling between aggressive, cautious, and learned Gaussian-prior modes.

This demonstration deploys the entire DA-MCTS [1] stack onboard a physical F1TENTH car in real time. The stack pairs a particle filter localizer with DA-MCTS as a unified planner and controller that produces velocity and steering commands using estimated pose. We compare against pure-pursuit, Stanley and MPC baselines that track a precomputed optimal raceline and Model Predictive Path Integral (MPPI) that is a sampling based method, and through experiments on an indoor racetrack we show that DA-MCTS computes the raceline online and produces competitive lap times, without any retuning.

## II. Implementation Details

The DA-MCTS pipeline is depicted in fig. 1, where each component runs as a separate ROS2 node pinned to a CPU core on the Jetson Orin Nano.

**Localization.** We use SynPF [2] for localization. It consumes the odometry readings and the LiDAR scan (topics `/odom` and `/scan`) and outputs the vehicle pose, the $x, y$ position and the heading $\psi$ in the global frame, which is published on `/pose` topic. Localization is event-triggered by the incoming LiDAR scan, which arrives at $40$ Hz, and our C++ implementation of SynPF has a WCET of $15$ ms on the Jetson Orin Nano.

**DA-MCTS Controller.** DA-MCTS plans by running Monte Carlo Tree Search with double progressive widening over a continuous action space, discovering a racing line online. DA-MCTS relies on two learned models. The dynamics model is a Sparse Variational Gaussian Process (SVGP), trained on data collected by driving the car along a precomputed raceline; it predicts the error of an extended-kinematic vehicle model and adds it back as a correction, and it takes cornering stiffness as an input feature so it generalizes to varying friction conditions. The second is an action-prior policy, a Gaussian process also trained on data collected along a precomputed raceline, which proposes context-appropriate actions for the search to sample. For further implementation details, see [1]. The defining idea of DA-MCTS is to adapt to a changing compute deadline by treating that deadline as an input to the search. Each planning step is given a time budget, and a small network reads how much of the budget remains, along with the search progress so far, to decide how the next action is sampled. We partition the continuous action space into three modes—aggressive, cautious, and the learned prior, shifting the sampling probability toward cautious actions when the deadline is small and toward aggressive maneuvers otherwise. The intuition is that with less compute it is safer to favor actions that slow the car, leaving more time to avoid a crash. This adaptation lets the planner stay safe under tight, fluctuating budgets of an embedded processor with no retuning. At each step DA-MCTS integrates the first action of its best trajectory into a velocity and steering command, published on `/drive` topic. The controller runs at a constant rate of $25$ Hz, which matches with the planning time step of MCTS rollouts and simulation steps.

## III. Experimental Results

Figure 2a shows the scaled indoor racetrack, obtained by mapping the area around our lab with SLAM Toolbox [3]. Figure 2b shows the standard F1TENTH platform used in the

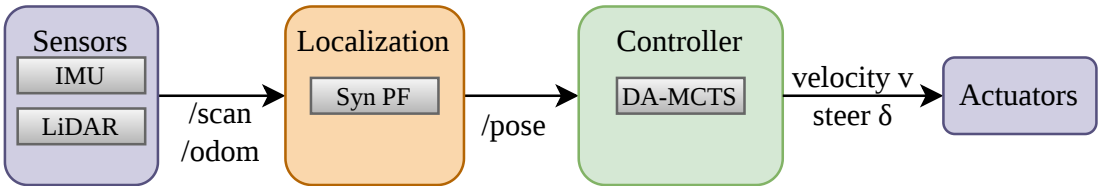
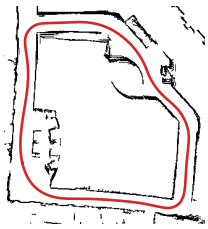
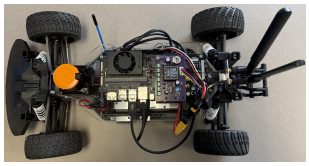



Fig. 1. **DA-MCTS Pipeline.** The SynPF particle-filter localizer fuses the LiDAR scan with wheel and IMU odometry to estimate the vehicle's global pose. The DA-MCTS controller uses this pose to plan online and produce velocity and steering commands directly to the actuators.

demonstration. The vehicle specifications include a Hokuyo UST-10LX 2D LiDAR, a VESC 6 MkVI that supplies motor commands and wheel odometry, integrated with an IMU, and an NVIDIA Jetson Orin Nano for onboard compute.



(a) Test track



(b) F1TENTH platform

Fig. 2. Demonstration setup. (a) The scaled indoor test track, mapped around our lab; the red line is the reference raceline. (b) The F1TENTH vehicle.

Table I reports the fastest lap by each method completed on the indoor lab track, in the F1TENTH ROS2 simulator and on the real car. We compare DA-MCTS against four baselines. Pure pursuit is a geometric tracker that steers toward a lookahead point on the precomputed reference raceline. Stanley tracks the same raceline using a front-axle cross-track and heading error. MPC is an iterative method with a SVGP as a dynamics model that optimizes over a horizon to follow the precomputed raceline. MPPI is a sampling based planner that rolls out many action sequences online rather than tracking a fixed raceline. DA-MCTS, our method, plans and controls online from the reward alone, discovering its trajectory with no precomputed raceline. MPC and MPPI fail to complete a lap as single MPC step takes 81 ms and an MPPI step 500 ms, far exceeding the 40 ms control deadline, so both miss the deadline and crash. The key result is that DA-MCTS plans online without a reference raceline and produces lap times close to the pure-pursuit and Stanley baselines that depend on a precomputed raceline as shown in table I. On the real car, DA-MCTS achieves the fastest lap, as it plans over a horizon using the learned dynamics model to anticipate and avoid imminent crashes, making it robust to sensor and localization noise.

## IV. Conclusion

We demonstrated an F1TENTH racing stack that runs Deadline-Aware MCTS in real time on an embedded Jetson Orin Nano, using a particle filter localizer and DA-MCTS as a deadline-aware controller. Our results show that the trajectory generated by DA-MCTS discovers a sub-optimal race line and attains lap times competitive with pure-pursuit and Stanley baselines that are given the optimal racing line. Future work will focus on extending the DA-MCTS planned trajectories to avoid obstacles, on improving lap times after each lap to discover the true racing line online, and on adding run-time assurance to keep the vehicle safe at all times.

TABLE I
Fastest lap time (s) on indoor race track. Lower is better. Baselines track a precomputed raceline; DA-MCTS plans online.

| **Method** | **Simulation (s)** | **Real car (s)** |
|---|---|---|
| Pure Pursuit [4] | 20.22 | 36.25 |
| Stanley [5] | 20.25 | 30.06 |
| MPC [6] | Crash* | Crash* |
| MPPI [7] | Crash* | Crash* |
| **DA-MCTS (ours)** | 20.46 | 26.45 |

* Crash = method failed to complete a lap.

## Acknowledgment

This project was funded in part by a Siemens Research Student Fellowship and in part by the National Science Foundation Grant 2238815. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Siemens Research & Predevelopment (RPD) and/or the National Science Foundation. The computing support for the work was also provided by NSF Chameleon.

## References

[1] S. Gupta, M. Yuhas, and A. Dubey, "DA-MCTS: Deadline-aware action exploration for safe real-time planning," in *Proceedings of the 32nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. Qingdao, China: IEEE, 2026.

[2] T. Y. Lim, E. Ghignone, N. Baumann, and M. Magno, "Robustness evaluation of localization techniques for autonomous racing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024.

[3] S. Macenski and I. Jambrecic, "Slam toolbox: Slam for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, 2021.

[4] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm."

[5] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, and Hoffmann, "Stanley: The robot that won the DARPA grand challenge," *Journal of Field Robotics*, 2006.

[6] A. Jain, M. O'Kelly, P. Chaudhari, and M. Morari, "Bayesrace: Learning to race autonomously using prior experience," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 155. PMLR, 2021.

[7] T. Kim, H. Lee, and W. Lee, "Physics embedded neural network vehicle model and applications in risk-aware autonomous driving using latent features," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 4182–4189, ISSN: 2153-0866.