Robustness Evaluation of Localization Techniques for Autonomous Racing

Tian Yi Lim*, Edoardo Ghignone*, Nicolas Baumann*, Michele Magno
Center for Project-Based Learning, ETH Zürich
{tialim, eghignone, nibauman, magnom}@ethz.ch

***Abstract*—This work introduces *SynPF*, an MCL-based algorithm tailored for high-speed racing environments. Benchmarked against *Cartographer*, a state-of-the-art pose-graph SLAM algorithm, *SynPF* leverages synergies from previous particle-filtering methods and synthesizes them for the high-performance racing domain. Our extensive in-field evaluations reveal that while *Cartographer* excels under nominal conditions, it struggles when subjected to wheel-slip—a common phenomenon in a racing scenario due to varying grip levels and aggressive driving behaviour. Conversely, *SynPF* demonstrates robustness in these challenging conditions and a low-latency computation time of $1.25\,\mathrm{ms}$ on on-board computers without a GPU. Using the F1TENTH platform, a 1:10 scaled autonomous racing vehicle, this work not only highlights the vulnerabilities of existing algorithms in high-speed scenarios, tested up until $7.6\,\mathrm{m\,s^{-1}}$, but also emphasizes the potential of *SynPF* as a viable alternative, especially in deteriorating odometry conditions.**

***Index Terms*—Autonomous Racing, Localization, Particle Filter, Monte Carlo Localization, SLAM, Robustness**

I. Introduction

Localization approaches for autonomous racing, such as pose-graph based Simultaneous Localization and Mapping (SLAM) [1] and Monte-Carlo Localization (MCL)-based (also called Particle Filtering, or PF) methods [2–4] depend on both *exteroceptive* and *proprioceptive* inputs. For example, LiDAR sensors offer range measurements for *exteroceptive* sensing, enabling the robot to perceive its environment. In contrast, *proprioceptive* measurements provide insight into the robot's internal states, processing signals from IMUs and wheel-odometry. Consequently, SLAM algorithms can map environments while localizing the robot. On the other hand, MCL-based techniques, relying on both sensing modalities and a pre-existing map, solely determine the robot localization using MCL [3, 4].

This paper leverages the F1TENTH autonomous racing platform to compare the performance of pose-graph optimization-based SLAM algorithms against MCL-based algorithms, considering the wheel-odometry signal quality to evaluate the performance in terms of robustness and latency. The robustness measurements can be qualitatively evaluated through scan alignment, the latter latency concern is quantitatively measurable, ultimately leading to a scan matching computation time of $1.25\,\mathrm{ms}$. Specifically, we compare the state-of-the-art (SOTA) pose-graph based SLAM algorithm *Cartographer* [1] with the newly proposed *SynPF*. The latter builds, combines, and synergizes upon prior MCL-based techniques [3, 4].

*Contributed Equally

II. SynPF Implementation Details

A commonly used motion model for MCL-based methods is the differential drive (diff-drive) model [2] which approximates the steering geometry of a car. This approximation, however, generates unrealistically high angular uncertainties at high speed, resulting in particles being in infeasible positions, reducing *particle efficiency*. An improvement used in *SynPF* is described in [4] (hereafter referred to as the TUM PF) which improves this approximation at high longitudinal velocities. This model considers the reduced lateral action space at high velocities, leading to a more realistic pose distribution for high-speed racing. This is motivated by the observation that at high velocities, for instance on straight sections of a racetrack, the car's angular uncertainty is low, as the steering input cannot be large. The improvement in the diff-drive model is further illustrated in fig. 1.

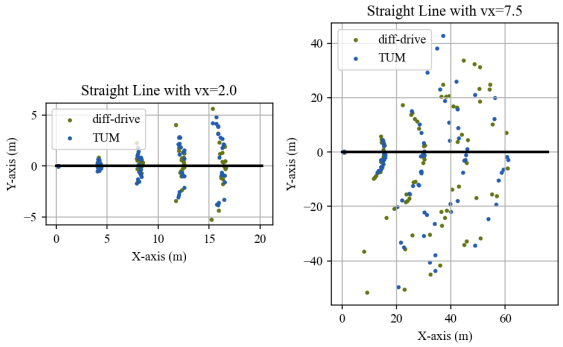

Fig. 1: Comparison of poses generated by diff-drive [2] and TUM motion models [4]. The left figure shows that at slow speeds, the two models are very similar. The right figure highlights how the TUM motion model further accounts for the reduced steering capacity at higher speeds.

The TUM PF [4] also proposes a *boxed* LiDAR layout. With a boxed layout, scanlines are chosen such that their intersections with a corridor of configurable aspect ratio are uniformly spaced. This is motivated by the observation that race tracks are corridor-like environments. By spacing the scanlines in this way, the boxed scanlines point further ahead down the racetrack, giving more information on the geometry of the racetrack further ahead. This results in more information with a constant number of scanlines.

In addition, a large proportion of computation effort in MCL methods is in evaluating the expected sensor range at a given pose. This is accelerated by using the `rangelibc` [3] library, which offers two modes of interest. Firstly, it allows the use of an available GPU to parallelize a ray-casting operation over all selected scanlines. This vastly speeds up the calculation of the expected ranges. Secondly, it offers a lookup table (LUT) option to pre-calculate all expected ranges for a discretized set of poses in the map. A 3D array is constructed, with entries for each possible x, y position of the LiDAR and orientation θ of the scanline. This results in constant-time query speed at the expense of memory usage. Utilizing one of the two methods greatly increases the rate at which the sensor model can be evaluated.

III. Experimental Results

Fig. 2 depicts the test track that was used to quantitatively evaluate the performance of either *Cartographer* or *SynPF*, with respect to the quality of the odometry input. The grip level was measured by pulling the test car laterally along the center of mass. As the test track features a surface with high friction (26 N nominal conditions), the car's tires were altered such that they feature significantly reduced static friction (19 N slippery conditions), by applying tape to the tires. This allows for mimicking the odometry quality degradation similar to a slippery floor. To isolate the odometry degradation effect, 10 laps were completed at the same speed scaling in both settings, where the lap time, scan alignment, and lateral deviation serve as proxy measurements for localization accuracy. The processor unit on the cars used was an Intel NUC on-board computer with an `i5-10210U` processor, without a dedicated GPU. Therefore, the LUT option in `rangelibc` [3] was utilized for this experiment.

Fig. 2: Test track used for quantitative localization accuracy evaluations. Left: Utilized test track, featuring grippy and high-quality odometry. Right: Racecar with taped and slippery tires, featuring low-quality odometry on the same test track.

Tab. I holds the resulting accuracy proxy measurements for *Cartographer* and *SynPF* under the influence of varying odometry quality. *SynPF* shows an approximately 0.15 s faster lap time in the low-quality odometry scenario and a significantly lower lateral deviation of 7.68 cm as opposed to *Cartographer*'s 11.43 cm error. On the other hand, when performing the same experiment on the test track under favourable grippy (high-quality) odometry conditions, tab. I showcases that the high-quality odometry signal allows *Cartographer* to perform a roughly 0.02 s faster average lap time than *SynPF* with a lower lateral error of 6.86 cm, as opposed to *SynPF*'s 8.22 cm error.

TABLE I: Lap time and computation results on the test track, featuring slippery (low-quality: LQ) and grippy (high-quality: HQ) odometry input. The average lateral error is with respect to the ideal race line. The scan alignment score is computed by the average percentage of overlapping scans and the track boundary. The compute metric refers to `htop` percentage of CPU core utilization.

Method	**Odom**	**Lap Time [s]**		**Error [cm]**		**Scan Align [%]** ↑	**Load avg** ↓
		$\mu \downarrow$	$\sigma \downarrow$	$\mu \downarrow$	$\sigma \downarrow$		
Cartographer	HQ	**9.167**	**0.097**	**6.864**	**0.264**	69.357	4.2
	LQ	9.428	0.126	11.432	**1.134**	61.710	
SynPF	HQ	9.184	0.153	8.223	0.406	**80.603**	**2.17**
	LQ	**9.280**	**0.093**	**7.686**	1.179	**79.924**	

Interestingly, for *Cartographer* the lateral error increases by 67% (6.86 cm to 11.43 cm) when going from high to low-quality odometry, while for *SynPF* the change is merely 6.9% from 8.22 cm to 7.68 cm.

IV. Conclusion

This work introduced *SynPF*, an MCL-based localization algorithm for autonomous racing that synergizes the efforts made in previous MCL-based approaches [3, 4]. The novel algorithm integrates a motion and sensor model that more accurately describes the high-speed racing scenario. *SynPF* ultimately yields a 1.25 ms, low-latency performance on on-board computers where GPUs are not available, and accuracy that rivals the performance of pose graph-based SLAM methods. Further, extensive in-field testing revealed that while under nominal grip levels the racing performance of a pose-graph SLAM method such as *Cartographer* is superior, with low-quality odometry signal the MCL-based *SynPF* shows robustness (-0.08% scan alignment and -6.9% lateral error) whereas the SLAM counterpart significantly worsens (-11.0% scan alignment score and +66.6% lateral tracking error). This allows operators to determine a priori to a race which kind of localization algorithm would be most suited for the given case.

References

[1] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278, doi: 10.1109/ICRA.2016.7487258.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005, ISBN 0262201623.

[3] C. H. Walsh and S. Karaman, "Cddt: Fast approximate 2d ray casting for accelerated localization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3677–3684, doi: 10.1109/ICRA.2018.8460743.

[4] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, "Ros-based localization of a race vehicle at high-speed using lidar," *E3S Web of Conferences*, vol. 95, p. 04002, 01 2019, doi: 10.1051/e3sconf/20199504002.