ROS-based localization of a race vehicle at high-speed using LIDAR

Tim Stahl[1] ,*Alexander* Wischnewski[2], *Johannes* Betz[1], and *Markus* Lienkamp[1]

[1]Chair of Automotive Technology, Technical University of Munich, Germany
[2]Chair of Automatic Control, Technical University of Munich, Germany

Abstract. An approach for LIDAR-based localization at high speeds is presented. In the proposed framework, the laser pose estimation is treated as a parallel redundant information, which is fused in an adjacent Kalman filter. The measurement and motion update step of the ROS-based adaptive Monte Carlo localization package is modified, in order to meet the requirements of a high-speed race scenario. Thereby, the key focus is on computational efficiency and the adaptation to characteristics arising at high speeds and at the limits of handling. An introspective performance evaluation monitors the position estimation process and labels generated outputs for adjacent components accordingly. The effectiveness of the proposed algorithm is illustrated in a real world high-speed experiment, autonomously driving a race vehicle – the *DevBot* – in a typical race environment.

1 Introduction

Racing in motorsports is a complex and demanding task. Even professional human drivers reach their cognitive and executive limits during a regular competition. Within the race series *Roborace* [1], the team of the Technical University of Munich aims to equip a race vehicle with the required software components enabling it to drive autonomously in a competitive manner. In an initial public event the *DevBot* (Figure 1) drove autonomously 150 km/h at the E-Prix event in Berlin [2]. The vision of this project is to transfer the gained knowledge from a race setting to automated urban traffic [3].

In order to drive fast and safe, an autonomous car needs to know accurately where it is located relative to its planned path and obstacles. Technically speaking, the task is called localization and describes the process of determining an agents pose relative to its surrounding environment. Several techniques serve this purpose, each coupled to their individual advantages and disadvantages. State of the art technologies include GPS/GLONASS, odometry information and optical localization.

In this paper, we focus on the latter one using LIDAR sensors. However, when pursuing a laser based approach for localization at high velocities on a race track, the following challenges have to be addressed:

- Computation time has to be low in order to maintain a decent update frequency (large distances traveled in short periods)
- Long straights on a race track provide few unique features for the LIDAR sensors and algorithms

Figure 1. *DevBot* - The development vehicle of *Roborace* holding various sensors and control units.[1]

- Uncertain motion model at high speeds and high lateral forces
- Inaccurate localizations should be detected and handled swiftly in order to react properly during high-speed segments

We present an extended particle filter based localization algorithm as well as an interfacing structure used for introspective performance evaluation. Our main contribution in this paper is the extension of the ROS[1]-based Adaptive Monte Carlo Localization (AMCL) algorithm, in a way to cope with high speeds on a race track, introspective status indicators and the performance evaluation on a real

[1]Robot Operating System [4]

race vehicle. Within this process, we reworked the motion and measurement update step of the particle filter to improve high speed localization on a race track.

The rest of the paper is organized as follows. The subsequent Section 2 covers related localization approaches in the field of mobile robotics and autonomous vehicles. The overall localization approach embodied in the *DevBot* is covered within Section 3. Key features of our LIDAR based localization approach are tackled in Section 4. Experimental results of the presented approach are given in Section 5. A discussion and concluding remarks are provided in Section 6 and Section 7, respectively.

2 Related Work

Automated vehicles require a robust and precise localization method. Especially when moving at high speeds, it is essential to position oneself precisely relative to the environment and to the planned trajectory. Several sensors and algorithms can be used to provide pose estimates. This paper focuses on laser based localization techniques.

Even within the domain of LIDAR based localization, a vast variety of techniques have been proposed. One prominent strategy is to rely on reflectivity variances of the ground plane. Structural features like pavement variations and lane markings are used for mapping and localization. Levinson et al. [5] initially proposed an approach based on ground reflectivity paired with a 3D LIDAR and IMU. Several modified versions followed while relying on a 2D laser only [6], being robust to slight changes in ground appearance [7] or handling weather variations like snow fall and wet roads by relying on Gaussian mixture maps [8, 9]. However, these approaches rely on lane markings and structural differences in the ground plane. When facing racetracks in the Formula-E series, the tracks often do not offer such features (e.g. a track is layed out by concrete walls in a city or an empty space).

Another group of approaches use 3D LIDAR sensors and focus on striking landmarks or objects in the environment. Schlichting et al. [10] save pole-like objects and planes in a feature map, which serves for localization landmarks in preceding runs. Wang et al. [11] implement an curb detection algorithm and use this information fused with GPS for localization. Obviously, those algorithms require these specialized objects to be present in the dedicated environment, which is not by default the case for race tracks. Furthermore, most of the approaches have only been tested at moderate speeds (e.g. 26.9 km/h on average) and require medium to high computational time (ranging from 50 ms to above 100 ms). Based on these facts it is questionable how these approaches would translate to high speeds.

To the best of the authors knowledge, only a few approaches address LIDAR localization at high speeds. Daoust et al. [12] propose a high-speed localization approach for railed vehicles up to a speed of 70 km/h. The approach is specifically tailored to a subway application and hardly applicable to a road scenario. Other approaches addressing high-speed agents [13**?**] consider the term "high-speed" in the robotic domain, which is the case for robots moving at 4 m/s.

Monte Carlo localization, first introduced by Fox et al. [14] is a particle filter based localization approach broadly used for several years now. In the past years various improvements and adaptations have been presented. Kümmerle et al. [15] extended the framework to utilize multilevel surface maps, which enables the robot to represent multiple levels in the environment. Recently, Bedkowski et al. [16] introduced a promising approach for parallel particle filter evaluation in order to process 3D maps and LIDAR sensors. However, none of the inspected approaches designed or evaluated their method at higher speeds than 75 km/h. Bedkowski et al. [16] noted that "it is evident that such numbers of calculation determine the applicability by only assuming the slow-motion robot equipped with the powerful GPU". An extended, adaptive version of the basic Monte Carlo localization algorithm is throughly presented in the book "Probabilistic Robotics" [17]. The version described in the book has been adopted by the ROS community [18] and is utilized on many mobile robots all over the world. Exemplary mentioned, Zaman et al. [19] as well as An et al. [20] use ROS packages – including Monte Carlo localization – for indoor navigation of a Pioneer 3-DX Robot and a custom mobile platform, respectively.

The proposed method for high-speed LIDAR localization of a race vehicle is based on an adaptive Monte Carlo approach, which will be revisited in Section 4.1. The extension is based on the open source package published in the ROS-community. We improved the motion update and measurement step in the particle filter to support vehicles at high speeds. Furthermore, a real world verification of the proposed method is provided.

3 Localization Framework

A vast variety of sensors can be used to generate a localization estimate of an autonomous system. Figure 2 gives a rough overview of a set of commonly used devices for localization. While some sensors provide position (or N-time derivative) estimates directly, optical based approaches require an algorithmic processing in order to generate an estimate. The proposed overall approach is based on a parallel, redundant information flow framework. As envisioned in Figure 2, various position estimates are generated and selectively fused in a Kalman filter. The benefit of this parallel structure is the independence of an specific sensor and the exploitation of the sensors' individual strengths.

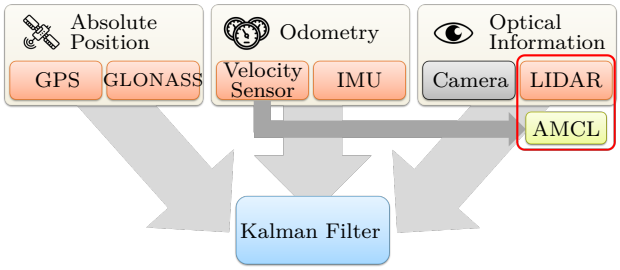

Figure 2. Sketch of available sensors used for localization. All sensors shown in *red* are included in our setup. This paper focuses on the LIDAR and AMCL, highlighted with a *red frame*.

In the wake of this, the overall localization estimation can cope with a GPS signal loss or a temporarily unavailable LIDAR estimate. This way, only the highly unlikely event of several systems being inactive would cause an emergency stop. Additionally, based on the parametrization of the Kalman filter's covariances, the framework can adapt to individual characteristics of every single estimator.

This paper focuses on LIDAR localization based on a particle filter (AMCL), as highlighted by the *red box* in Figure 2. Since the LIDAR estimate should serve as a redundant information, a requirement is to not use GPS measurements in the motion update at all. The characteristic of the LIDAR estimation – performing very accurate in the lateral direction and only decently in the longitudinal direction (Section 5) – is considered in the Kalman filter parametrization. Thereby, the trust in the lateral information is rated higher than the longitudinal estimate.

4 High-Speed LIDAR Localization

The presented approach builds upon AMCL, a popular particle filter based localization framework. Section 4.1 revisits the basic functionality and its components. In order to keep track of the vehicles position estimate at high speeds, several modifications of the basic algorithm were introduced (Section 4.2). The most crucial goal is to provide a reliable overall position estimate at any time. In the wake of this, the system should detect localization degrades introspectively and label the generated estimates correspondingly. The module providing this functionality is elaborated in Section 4.3. One of the most crucial parameters – the maximum amount of particles – is investigated and adapted to the target hardware in Section 4.4.

4.1 Basic Adaptive Monte Carlo Localization

Monte Carlo localization (MCL) is a localization algorithm based on a particle filter described by [17]. The method uses a particle filter to converge to the most probable pose within a given occupancy grid representation of the environment (Algorithm 1). In

Algorithm 1 Monte Carlo Localization

procedure MCL($\mathcal{X}_{t-1}, u_t, z_t, m$)
  $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
  for $m = 1$ to M **do**
    $x_t^{[m]} =$ **sampleMotionModel**$(u_t, x_{t-1}^{[m]})$
    $w_t^{[m]} =$ **measurementModel**$(z_t, x_t^{[m]}, m)$
    $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \left\langle x_t^{[m]}, w_t^{[m]} \right\rangle$
  for $m = 1$ to M **do**
    draw $x_t^{[m]}$ from $\bar{\mathcal{X}}_t$ with probability $\propto w_t^{[m]}$
    $\mathcal{X}_t = \mathcal{X}_t + x_t^{[m]}$
  return $\mathcal{X}_t$

this setup, each particle m represents a potential pose x_t^m, i.e. a hypothesis of the agents position and orientation in the grid. Initially at time step $t = 0$, the particles $\mathcal{X}_{t=0} = \{x_0^1, x_0^2, ..., x_0^M\}$ are spread randomly in a predefined region, i.e. the whole map or a certain spot on the map. In other words, the algorithm is not sure about the actual pose and distributes several random guesses. Whenever the agent moves (action u_t), all the existing particles are shifted along while introducing some noise. The amount of noise added is described in an underlying odometry model. Thereafter, every incoming LIDAR measurement z_t is evaluated by utilizing recursive Bayesian estimation. Within this process, the measurement z_t is transformed into each of the particles poses and evaluated with respect to the degree of compliance between map and measurement. Particles are re-sampled according to the correlation scores w_t^m, i.e. more particles are generated in the region of high weights. After several update iterations, the particles should converge towards the actual position of the vehicle.

MCL has been improved by using an adaptive amount of particles. The number of particles used within AMCL is then determined by an error estimate. Due to this improvement, the algorithm uses many particles whenever the estimation performance is poor and saves computation effort by using few particles whenever the particles converged nicely. Readers interested in further insights to this topic are referred to [17]. When facing a high-speed scenario one has to take special care of this issue, which will be further investigated in Section 4.4.

4.2 Modifications Monte Carlo Localization

Besides resampling, the two major steps in the particle filter are the motion and measurement updates. We investigate both of them and introduce improvements regarding a race scenario.

4.2.1 Measurement Update

One of the most crucial issues when facing high speeds is the calculation time. The longer the calculation takes, the further the vehicle travels in the meantime.

Since the position is calculated based on a LIDAR stream captured at the start of the calculation period, the position estimate is always outdated to a certain extent.

In order to match the calculation cost of the measurement step (Algorithm 1-5) to the used hardware, the ROS-package provides several parameters, which influence the computational load. Probably the most trivial option is to limit the number of particles M used in the filter (Section 4.4). Another option is to reduce the amount of LIDAR beams to be processed in the sensor update. Especially when using multiple and/or high-resolution LIDAR sensors it is infeasible to evaluate every single beam, since – depending on the map resolution – no further information is gained. Commonly, only a representative subset of all the LIDAR measurements is evaluated, since this is sufficient to generate a matching score for each particle pose. The stock package provided by ROS samples a parameterizable amount of entities at an equal angular displacement from the full measurement set.

Since a race vehicle is commonly driving in a sort of corridor with walls to the sides and free space ahead of the vehicle, this approach is not the most efficient. A simplistic extracted 360 deg LIDAR beam model, hosting the constant angular displacement in the before mentioned environment, is illustrated in Figure 3. Most of the extracted beams hit the wall right next to the vehicle in a similar spot, which does not provide a substantial information gain. On the other hand, only few information is gained at the far end ahead and behind the vehicle due to a sparse coverage with LIDAR beams. Following the default model, the only chance to gain more information is to increase the number of extracted entities, which results in a higher computational load. We introduce an approach to extract a corridor optimized pattern of measurements. The goal is to increase the information gain, while keeping the amount of extracted LIDAR beams constant. For this purpose, the angular displacement is calculated based on equal distances on the borderline of a surrounding rectangle with parameterizable aspect ratio. In order to envision this effect more clearly, the two left-hand plots in Figure 3 hold only 10 extracted beams. One can clearly see that the default approach primarily extracts information of beams in the close proximity to the origin, whereas the boxed approach is evenly spread in this corridor example. When increasing the number of extracted beams to 30 (default value of the ROS AMCL implementation), a similar density distribution unfolds (Figure 3, right).

The method introduced in this paper determines the best angular displacement for each single beam, given a number of beams to be extracted N_{out} as well as the aspect ratio of the bounding box. The routine calculates the surrounding distance of an rectangle with the shape defined by the rectangle and divides it into N_{out} equal distance segments. Afterwards,

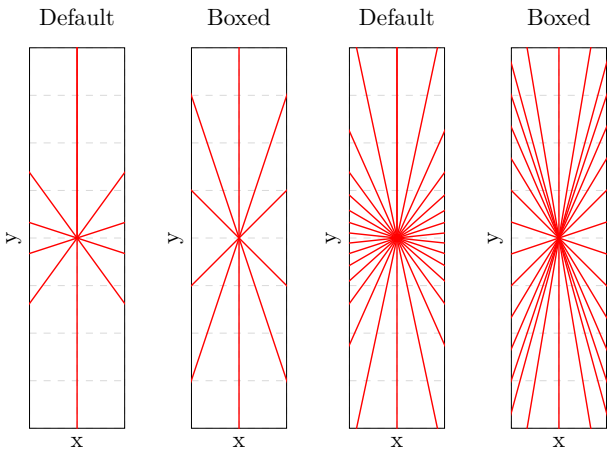

Figure 3. Comparison of default and introduced boxed approach for LIDAR beam extraction. Here shown for a corridor with an aspect ratio of 4:1. The two plots on the *left* simulate an extraction of 10 beams, whereas the two plots on the *right* hold 30 beams. It can be seen, that with the default approach several beams hit the wall right next to the origin and only provide sparse information in the far front and rear region. In this sketch a 360 deg LIDAR is assumed as extraction basis, with an angular increment of 0.25 deg (Corresponding actual *minimal* increment, when fusing multiple *Ibeo ScaLa B2*® LIDARs).

temporary coordinates along the outline of this box are calculated. In order to obtain a symmetrical pattern, the first beam is always placed in the front center of the vehicle. Each temporary coordinate (x, y) is converted into an angle in the LIDAR's frame. Based on this calculation, the algorithm returns the indexes $\mathcal{I}$ of the physical LIDAR beams closest to those calculated angles. It should be noted, that this calculation only has to be performed once, since every consecutive LIDAR evaluation uses the same angular pattern.

4.2.2 Motion Update

The motion update step in the ROS version of AMCL is tailored to indoor robotic agents. Most of these platforms provide omni-directional or differential drives. These vehicle dynamics differ from car-like dynamics and are not suited to high velocities by default. The approach presented in this work is based on the differential drive model, since it is the most similar to car like dynamics. Thrun et al. [17] presented a basic probabilistic odometry model (Algorithm 2), where each movement is represented by three steps: an initial rotation δ_{rot1}, a translational shift δ_{trans} and another rotation δ_{rot2} at the translated pose (Figure 4). These step offsets are calculated based on the last odometry reading $u_t = (\bar{x}_{t-1}, \bar{x}_t)^T = ((\bar{x}, \bar{y}, \bar{\theta}), (\bar{x}', \bar{y}', \bar{\theta}'))^{\mathrm{T}}$ and the last state[2] x_{t-1}, as shown in Algorithm 2-(2–4). The resulting values represent

[2]State variables without a bar refer to the particle filter's internal state representation, whereas a bar above the variable indicates odometry measures.

Algorithm 2 Sample Odometry Motion

1: **procedure** SAMPLEMOTIONMODEL(u_t, x_{t-1})
2: $\delta_{\text{rot1}} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
3: $\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$
4: $\delta_{\text{rot2}} = \bar{\theta} - \bar{\theta}' - \delta_{\text{rot1}}$
5:
6: $\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - n_{\mathcal{N}}(\alpha_1\delta_{\text{rot1}} + \alpha_2\delta_{\text{trans}})$
7: $\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - n_{\mathcal{N}}(\alpha_3\delta_{\text{trans}} + \alpha_4(\delta_{\text{rot1}} + \delta_{\text{rot2}}))$
8: $\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} - n_{\mathcal{N}}(\alpha_1\delta_{\text{rot2}} + \alpha_2\delta_{\text{trans}})$
9:
10: $x' = x + \hat{\delta}_{\text{trans}}\cos(\theta + \hat{\delta}_{\text{rot1}})$
11: $y' = y + \hat{\delta}_{\text{trans}}\sin(\theta + \hat{\delta}_{\text{rot1}})$
12: $\theta' = \theta + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}$
13: **return** $x_t = (x', y', \theta')^T$

the observed motion step and are the basis for individual particle motion entities $\left\langle \hat{\delta}_{\text{rot1}}, \hat{\delta}_{\text{trans}}, \hat{\delta}_{\text{rot2}} \right\rangle$, generated by summing each of them with a unique, zero-mean Gaussian sample $n_{\mathcal{N}}(\sigma) = x \sim \mathcal{N}(\mu = 0, \sigma)$ (parameterized via $\alpha_{\{1,2,3,4\}}$, see Algorithm 2-(6–8)). Finally, these values are translated back into poses by applying simple trigonometrical rules. Further details and a proof of this approach can be found in [17].

One of the key features of this motion sample step is to at least cover or better overestimate the whole area of expected maximal error in the provided motion step u_t. When moving a moderate speeds coupled with a proper parameter tuning, this introduced odometry approach performs well. However, when facing high velocities (i.e. larger motion steps δ_{trans}), more and more rotational noise is introduced (Algorithm 2-(6,8)). This spread in particles causes localization inaccuracies and a growing computational effort (due to the generation of further particles for adaptive filters).

Since the absolute extent of the expected noise originating from relative sensor inaccuracies scales with the available action space, we propose a velocity dependent angular noise generation. Road vehicles are limited in their available lateral tire force, which is easily exceeded when traveling at high speeds [21]. The centripetal acceleration a_c for simple point masses traveling on a circle with radius r is defined as:

$$a_c = \frac{v^2}{r}. \tag{1}$$

One can easily see, that the required lateral force rises quadratically with higher speeds, while keeping the track radius constant. The sum of longitudinal and lateral force may not exceed a certain ground and tire specific value. In the region of low speeds, this fact does not play an important role, since the traction forces will stay in their bounds. However, when facing higher velocities, the executable turn radius grows with the vehicles speed. In order to cope with this issue, we reduce the introduced rotational noise $\hat{\delta}_{\text{rot1}}, \hat{\delta}_{\text{rot2}}$ according to the current speed (i.e. with the translated distance δ_{trans}). The relation of approximated track

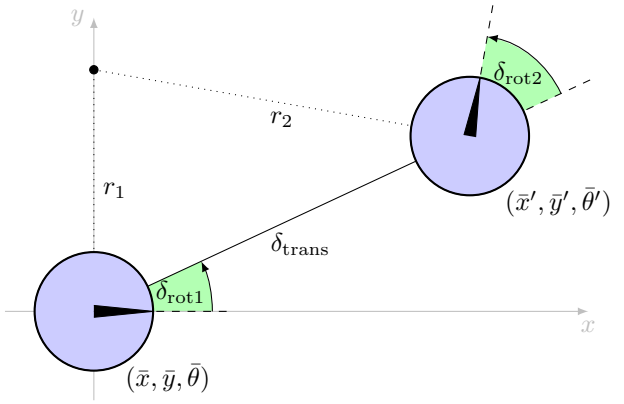

Figure 4. Geometrical relations of the utilized odometry model consisting of two rotation and one translation step. The blue discs indicate the vehicles start and end pose, where the black arrow represents the heading.

radius r and rotation change $\delta_{\text{rot}\{1,2\}}$ is found to be non-linear[3]:

$$r = \frac{\delta_{\text{trans}}\left(\sin\left(\frac{\pi}{2} - \delta_{\text{rot1}}\right) + \sin\left(\frac{\pi}{2} - \delta_{\text{rot2}}\right)\right)}{2\sin\left(\delta_{\text{rot1}} + \delta_{\text{rot2}}\right)}, \tag{2}$$

which can be approximated to

$$r \approx \frac{\delta_{\text{trans}}}{(\delta_{\text{rot1}} + \delta_{\text{rot2}})}, \tag{3}$$

especially when facing moderate angles ($< 30\,\text{deg}$).

Putting everything together, the maximum executable radius increases with the square of the executed velocity (assuming no longitudinal acceleration and ideally a constant maximum sustainable lateral force by the tire). The radius itself is reciprocally proportional to the sum of the angular components in the described motion model. In order to achieve a conservative and linear representation of this behavior, we reduce the angular noise of samples linearly with larger translated distances, i.e. higher velocities. Thereby, a threshold γ_{th} defines the minimum translation required for the mechanism to become active in scaling. The lines 6 and 8 in Algorithm 2 get replaced by the structure as follows:

$$\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - n_{\mathcal{N}}\left(\alpha_1\delta_{\text{rot1}} + \alpha_2\frac{1}{\max(\delta_{\text{trans}}, \gamma_{\text{th}})}\right), \tag{4}$$

here shown exemplary for δ_{rot1}.

In order to achieve an accurate lateral localization, parameterizable (α_5) lateral noise is added to each sample's pose. In contrast to lateral particle spread added via rotational noise, these entities maintain the same heading tendency. At this point it should be noted, that this lateral noise is chosen large on purpose in order to enable the filter to pick the optimal lateral

[3] Utilizing the law of sines for the triangle defined by the intersection of orthogonal lines on the initial and resulting velocity vector (r_1 and r_2 in Figure 4).

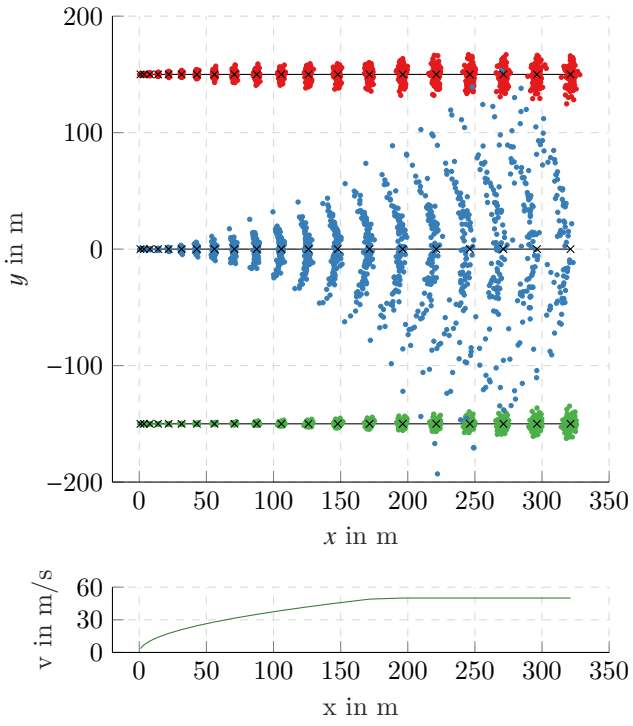

Figure 5. Comparison of differently parametrized position belief based on stock (*blue* and *green*) and proposed (*red*) odometry calculation for $N = 20$ time steps. The simulated incoming odometry measures from adjacent time steps are shown in *black*. The *green* line in the bottom plot displays the velocity along the course. For visualization purposes (distinguishable time steps), the longitudinal noise in the model was reduced significantly.

displacement in each update step. When following this strategy, we obtain particles with less angular spread at high speeds while maintaining a decent lateral coverage.

A comparison of two parameterizations of the stock update step and a parameterization of the proposed motion update step is shown in Figure 5. The two models were parameterized identically (the modified angular spread for the stock model was parametrized with proper angular noise for low velocities (*blue*) and high velocities (*green*)) and supplied with ideal pose measures ($v_{\max} = 50\,\mathrm{m/s}, a = 7\,\mathrm{m/s^2}$, update frequency $f = 2\,\mathrm{Hz}$, no heading variation). In this simulation, no measurement steps were performed. Therefore, the motion update was applied iteratively to each particle. When parameterizing the odometry model, one tries to fit or even overestimate the maximal error in the odometry measurement. In this example, the stock odometry model is tuned to suit the spread at low velocities (*blue*, i.e. a reasonable spread of the particles can be observed on the first 50 m) and at high velocities (*green*, i.e. a decent further spread of the particles can be observed while moving at $v_{\max}$). Thereby, it can be seen that the pose candidates based on the stock odometry model spread either to much at high speeds (*blue*) or introduce almost no variance at low speeds (*green*). In contrast, the spread of the proposed approach (*red*) reaches a steady spread at a certain velocity level while still offering variance at low speeds.

4.3 Introspective Performance Evaluation

Since a wrong localization estimate could cause severe accidents, it must be avoided by all means. In order to tackle this issue, we introduced several introspective watchdogs and mechanisms. The proposed methods check the generated location estimate for validity and publish a corresponding localization status s_{loc}. The adjacent Kalman filter then considers this information in the location estimate fusion. Along with the pose estimate, the proposed framework holds three status levels:

- 2 – proper functionality
- 1 – estimate of poor quality
- 0 – invalid location estimate

The status bit is composed of an initialization flag s_{init} (rising to 1 upon availability of all required components), a map dependent status s_{map} as well as a covariance evaluation s_{cov}.

$$s_{\mathrm{loc}} = \begin{cases} 2, & \text{if } s_{\mathrm{init}} \neq 0 \wedge s_{\mathrm{map}} \neq 0 \wedge s_{\mathrm{cov}} \neq 0 \\ 1, & \text{if } s_{\mathrm{init}} \neq 0 \wedge s_{\mathrm{map}} \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

When the particle filter is unable to properly match the perceived environment to the map representation, the odometry updates commonly carry the localization estimate outside the maps known free space. A common occupancy grid representation holds values for each grid cell m_i in the range of $[-1, 1]$, where -1 represents unknown space and 0 to 1 the probability of occupation. During execution, the algorithm reads the occupancy probability entry $p(m_i)$ of the grid cell m_i at the location of the pose estimate and returns a null status when exceeding the range $[0, \gamma]$. The constant γ is a parameterizable upper threshold.

$$s_{\mathrm{map}} = \begin{cases} 1, & \text{if } p(m_i) \in [0, \gamma] \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Another performance indicator is the variance of the pose estimation. This property correlates to the spread of the particles. The proposed method transforms the variances from map coordinates to the vehicles coordinate frame (relying on the current heading estimate). Thereafter, lateral (C_{yy}) and longitudinal (C_{xx}) variances are evaluated and rated against thresholds $\theta_{\{\mathrm{x,y}\}}$ individually. Since the pose estimation in long corridors performs better in lateral than in longitudinal direction, the thresholds are set accordingly.

$$s_{\mathrm{cov}} = \begin{cases} 1, & \text{if } C_{\mathrm{xx}} < \theta_{\mathrm{x}} \wedge C_{\mathrm{yy}} < \theta_{\mathrm{y}} \wedge C_{\mathrm{yawyaw}} < \theta_{\mathrm{yaw}} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

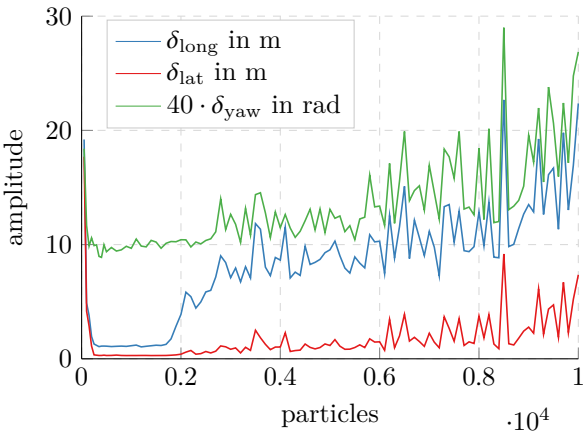

Figure 6. Mean lateral, longitudinal and angular displacement along one lap on a track, while varying the number of maximal particles used.

Figure 7. Areal footage of the test field with overlayed LIDAR scans (*red*) visualizing the test track (*blue*) framed by cones.

4.4 Particle Quantity

As mentioned in the subsections before, computation time is one of the most crucial topics when facing high speeds. When dealing with particle filters, the amount of particles used has a linear influence on the calculation time. For adaptive particle filters, one usually limits the maximum amount of particles used. When facing high speeds, setting the proper maximum amount of particles is a crucial step. A number chosen too high may result in a computation delay on the target hardware, which is self-enforcing. For example, a computation delay may result in an outdated estimate, the uncertainty raises, more particles are added and thereby even further delay is introduced. On the other hand, to few particles result in inaccurate localization results. In order to determine the proper dimension for the specific machine used, we simulated several track runs with varying maximal particle numbers on the target hardware (*NVIDIA DRIVE PX2*). For each run, the mean estimation error compared to the reference pose was evaluated. We evaluated the performance at an upper bound of $\{1, 50, \ldots, 500, 600, \ldots, 10000\}$ particles, while averaging 10 runs of each quantity step (resulting in a total of 1060 runs). Figure 6 visualizes the number of particles and the corresponding averaged lateral, longitudinal and angular estimation error for a complete simulated lap on the Formula-E race track in Hongkong. It was found, that the position estimation performance started to drop above 1900 particles (given the reason explained above) and performed good above a minimum of 300 particles. We picked an upper limit of 600 particles for our target machine, in order to be on the safe side with respect to dexterous situations (more particles needed) and the fact of sharing the computation power with other algorithms on the machine (less particles can be handled properly).

5 Experimental Results

The described method was evaluated on a real autonomous race vehicle, called *DevBot* (Figure 1). The vehicle is equipped with four *Ibeo ScaLa B2*® LIDAR sensors (two on the front wing facing forward and two on the sides facing slightly backwards). For global localization, the vehicle is equipped with an *OXTS 4000* inertial and position measurement unit, which is solely used for particle filter initialization within the algorithm. The odometry information is extracted from an optical, slip-free dynamics sensor (*Kistler Correvit*® *SFII P*).

Most of the tests were executed on a custom track layout marked by cones on an abandoned airfield (Figure 7). Furthermore, data recorded on various Formula-E race tracks was used for development and validation. Within this paper, we analyze an autonomously driven high-speed (> 150 km/h) run on the test track. One lap on the test track is about 970 m long. The discussed run includes 8 laps on the track, with lowered velocity and acceleration profile in the first two laps. The planned maximum lateral and longitudinal acceleration were both set to $8\,\text{m/s}^2$. The run was executed autonomously without a backup driver inside, but several emergency stop transmitters positioned around the track.

First, the actual performance on the track is evaluated in the following subsection. Second, the stock and modified localization approach is compared based on a bag record of the discussed run (Section 5.2).

5.1 Real World Run Analysis

The position logs of the localization estimation, odometry and GPS reference are visualized in Figure 8. It can be seen that the odometry position estimate is fairly good, but still drifts away several meters each lap. In contrast, the LIDAR based estimation sticks to the GPS reference.

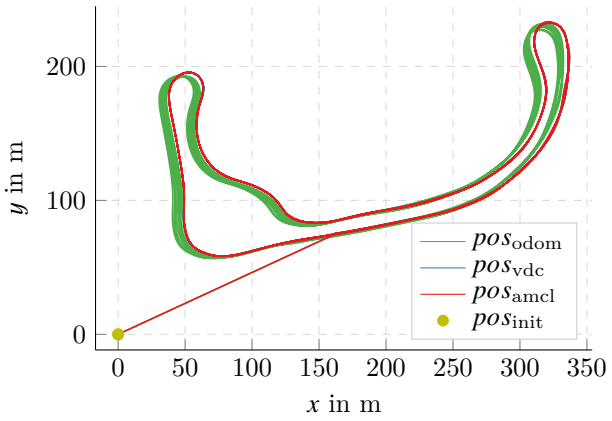

Figure 8. Position (x,y) log of the odometry output (*green*), GPS reference (*blue*) and AMCL position estimate (*red*). The *yellow* circle highlights positions, where a particle filter initialization was triggered.

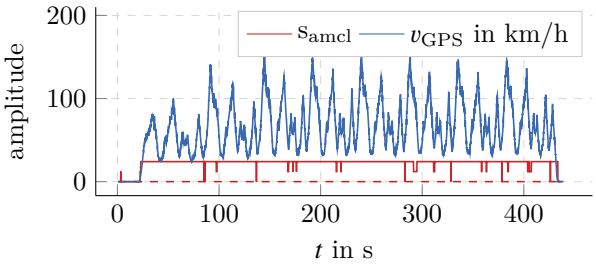

Figure 9. Temporal course of the vehicles velocity (*blue*), based on GPS measurements. The prediction status of AMCL is shown in *red*, where high amplitudes resemble the status 2 and the dashed reference line status 0.

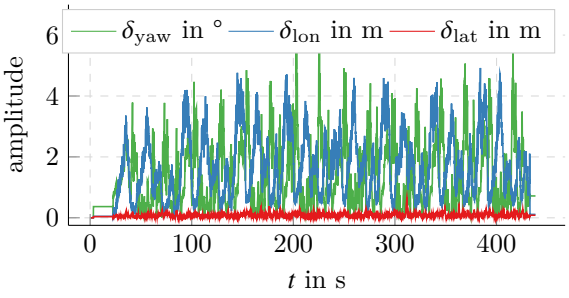

Figure 10. Absolute lateral (*red*), longitudinal (*blue*) and angular (*green*) offset of predicted AMCL pose compared to GPS reference.

The vehicles velocity throughout the run peaked > 150 km/h several times on the long straight, plotted *blue* in Figure 9. The average velocity for a flying racing lap settled at 72.6 km/h. All calculated metrics in the following are based on the data gathered during a driving phase ($v > 0$).

Figures 10 and 11 visualize the euclidean[4] and angular displacement as well as the covariances in these

[4]It should be noted that the euclidean displacement was split into lateral and longitudinal error, where the *estimated heading*

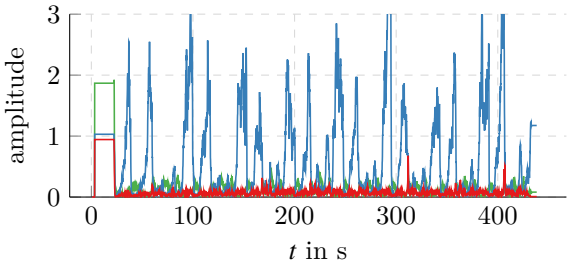

Figure 11. Temporal course of the squared covariance in the vehicles lateral (*red*), longitudinal (*blue*) and rotational (*green*, up-scaled by factor 200) component.

dimensions. It can be seen, that the longitudinal offset between GPS and AMCL estimation is relatively high on the long straights and reduces again at each corner. One possible cause for this is the feature similarity along the track. The feature patterns extracted on long straights or long bended curves resemble the LIDAR patterns extracted on various spots along the track. Furthermore, the temporal offset between time-stamp of LIDAR perception and finally estimated pose has an noticeable impact on the longitudinal component at high-speeds. For example, when traveling at a speed of 42 m/s and a LIDAR to estimation period[5] of 40 ms, the induced offset calculated to $42\,\mathrm{m/s} \cdot 0.04\,\mathrm{s} = 1.68\,\mathrm{m}$. While this offset is present in Figure 10, the overall framework offers a delay compensation, which is implemented in the adjacent Kalman filter. Displayed in numbers, the longitudinal error had a few peaks slightly below 5 m with an average absolute error of 1.96 m.

When analyzing the angular error, one can find a decent performance with maximal outliers of 5 deg. The peaks in the angular error mainly occur at sharp turns. Thereby, the temporal shift between LIDAR measurement and GPS reference may play a major role at locations of high curvature changes.

As mentioned in Section 3, the main focus of this approach is on the lateral localization. The lateral position accuracy is throughout the run good, with an average value of 0.086 m. Figure 10 shows a few peaks in the lateral offset (e.g. at 311 s), ranging up to 0.70 m. In these cases, the longitudinal error was still present at the entry of a corner – i.e. the pose estimate is a couple of meters behind the actual pose – and therefore caused a cutting behavior in that curve. Nevertheless, all of the peaks have been identified by the introspective performance rating. During these periods, the status was set to 1 and therefore not fused into the overall localization estimate. By utilizing the introspective status management as described in

of the vehicle was used as reference. Therefore, this calculation may be subjected to slight inaccuracies.

[5]While using 600 particles in the discussed run, the lower and upper 5% percentile of this temporal offset (incoming LIDAR message to estimated pose) settled at 19.3 ms and 28.7 ms, respectively. The actual offset will be higher due to sensor internal and transmission delays.

Table 1. Performance Comparison

Odometry Model	Boxed	Lateral Offset
stock - tuned low v	false	0.169 m
stock - tuned high v	false	0.179 m
proposed	true	0.111 m

Section 4.3, the mean absolute lateral error drops from 0.086 m (including the spikes) to 0.084 m (the peak error drops from 0.70 m to 0.45 m).

Overall, the introspective component holds a low false alarm rate and intervened for relatively short periods. In the driving period of the vehicle, only 2.96 % of the time resulted in a status other than 2 (*red* in Figure 9). It should be noted, that the status 0 flags in the status plot result from single false occupancy values in the grid map.

5.2 Comparison

In order to properly compare the proposed localization approach to the stock AMCL algorithm, both methods are subjected to an identical bag file of the previously discussed run. The bag file contains all relevant sensor messages and the occupancy grid of the race track. We used the same overall tuned parameters for all approaches, except the parameters of interest.

As discussed in Section 4.2.2, the stock odometry model is not designed to cover low and high velocities concurrently. Therefore, we compare the proposed algorithm against two parameterizations of the stock odometry model ("diff-corrected") as introduced in Figure 5. Thereby, two configurations are tackled: one realization is tuned to perform best at low and the other at high velocities. We averaged the lateral offset of five consecutive runs for each considered approach. In order to exclude influences by the number of maximal particles used, each run was parametrized with an individual upper limit of the set $600, 700, \ldots, 1000$ (reasonable values according to results in Section 4.4). Table 1 holds the results of this analysis.

It should be noted that the introspective performance evaluation was not active in any of the runs. That way one can compare the plain algorithms on itself. The proposed methods reduce the averaged lateral estimation error by more than 34%. One prominent observed reason for the higher mean errors of the stock approach are sporadic wrong localization estimates.

6 Discussion

While the lateral estimation performance of the proposed approach is good, the longitudinal estimation at high speeds should be further investigated. The limiting factors in this sense are the lack of unique features on long straights on a race track as well as the time delay between LIDAR capture and calculated pose estimate.

Furthermore, the proposed approach relies on a fairly accurate initial location guess (maximal offset of actual pose about 10 m). In this sense, we did not investigate the kidnapped robot problem [22], where many more particles would be required for an initial global pose estimation. In order to up the particle count, multi-core or GPU particle filters [23] promise further acceleration.

7 Conclusion

The presented approach allows LIDAR based localization at high speeds. The proposed framework is based on a parallel structure, fusing independent LIDAR and GPS position estimates in an adjacent Kalman filter. An open source adaptive particle filter – integrated into the ROS framework – is extended in a way to cope with high speeds. As highlighted in this work, computation time is the key limiting factor when hitting high velocities. In the wake of this, the measurement and motion update step of the particle filter is adapted to fit the race scenario requirements. In order to improve the gained information for a constant computation effort, a boxed LIDAR beam extraction method is proposed, designed to suit the corridor like structure of Formula-E race tracks. The motion update step is based on a new odometry model, taking the reduced available lateral action space into account. Additionally, the introduced introspective performance evaluation monitors the position estimation process and sends a corresponding status message to the adjacent Kalman filter. That way, wrong or inaccurate measures do not get fused into the overall position estimate. Finally, the framework is parametrized for a target vehicle and evaluated in a real world scenario.

Acknowledgment and Contributions

Tim Stahl initiated the idea of this paper and contributed essentially to its content. Alexander Wischnewski and Johannes Betz contributed to the conception and realization of this research. Markus Lienkamp made an essential contribution to the conception of the research project. He revised the paper critically for important intellectual content. He gave final approval of the version to be published and agrees to all aspects of the work. As a guarantor, he accepts the responsibility for the overall integrity of the paper.

We thank the Roborace team for giving us the opportunity to work with them and using their vehicles for our research. We express gratitude to TÜV-Süd and TUM for funding of the underlying research project.

References

[1] *Global championship of driverless cars*, `https://www.roborace.com/`

[2] L. Goudkamp, B. Rundfunk, *Roboter-rennauto vs mensch: Mit einem algorithmus beim roborace 2018 zum sieg?* (2018), `https://www.br.de/fernsehen/ard-alpha/sendungen/campusmagazin/robo-race-autonome-auto-roboter-rennen-berlin-challenge-100.html`

[3] J. Betz, A. Wischnewski, A. Heilmeier, F. Nobis, T. Stahl, L. Hermansdorfer, B. Lohmann, M. Lienkamp, chassis.tech 2018 (2018)

[4] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, **3** (2009)

[5] J. Levinson, M. Montemerlo, S. Thrun, *Map-Based Precision Vehicle Localization in Urban Environments*, in *Robotics: Science and Systems III* (Robotics: Science and Systems Foundation, 2007), ISBN 9780262524841

[6] I. Baldwin, P. Newman, *Road vehicle localization with 2D push-broom LIDAR and 3D priors*, in *IEEE International Conference on Robotics and Automation (ICRA), 2012* (2012), pp. 2611–2617, ISBN 978-1-4673-1405-3

[7] J. Levinson, S. Thrun, *Robust vehicle localization in urban environments using probabilistic maps*, in *IEEE International Conference on Robotics and Automation (ICRA), 2010* (2010), pp. 4372–4378, ISBN 978-1-4244-5038-1

[8] R.W. Wolcott, R.M. Eustice, *Visual localization within LIDAR maps for automated urban driving*, in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, Piscataway, NJ, 2014), pp. 176–183, ISBN 978-1-4799-6934-0

[9] R.W. Wolcott, R.M. Eustice, *Fast LIDAR localization using multiresolution Gaussian mixture maps*, in *2015 IEEE International Conference on Robotics and Automation (ICRA 2015)* (IEEE, Piscataway, NJ, 2015), pp. 2814–2821, ISBN 978-1-4799-6923-4

[10] A. Schlichting, C. Brenner, *Localization using automotive laser scanners and local pattern matching*, in *2014 IEEE Intelligent Vehicles Symposium Proceedings* (IEEE, 2014), pp. 414–419, ISBN 978-1-4799-3638-0

[11] L. Wang, Y. Zhang, J. Wang, IFAC-PapersOnLine **50**, 276 (2017)

[12] T. Daoust, F. Pomerleau, T.D. Barfoot, *Light at the End of the Tunnel: High-Speed LiDAR-Based Train Localization in Challenging Underground Environments*, in *CRV 2016* (2016), pp. 93–100, ISBN 978-1-5090-2491-9

[13] K. Lingemann, A. Nüchter, J. Hertzberg, H. Surmann, Robotics and Autonomous Systems **51**, 275 (2005)

[14] D. Fox, W. Burgard, F. Dellaert, S. Thrun, AAAI/IAAI **1999**, 2.2 (1999)

[15] R. Kümmerle, R. Triebel, P. Pfaff, W. Burgard, Journal of Field Robotics **25**, 346 (2008)

[16] J.M. Bedkowski, T. Röhling, Industrial Robot: An International Journal **44**, 442 (2017)

[17] S. Thrun, W. Burgard, D. Fox, *Probabilistic robotics* (MIT press, 2005), ISBN 0262303809

[18] *amcl - ros wiki*, `http://wiki.ros.org/amcl`

[19] S. Zaman, W. Slany, G. Steinbauer, *ROS-based mapping, localization and autonomous navigation using a Pioneer 3-DX robot and their relevant issues*, in *2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC)* (IEEE, Piscataway, NJ, 2011), pp. 1–5, ISBN 978-1-4577-0068-2

[20] Z. An, L. Hao, Y. Liu, L. Dai, International Journal of Innovative Science and Modern Engineering **5**, 47 (2016)

[21] H.B. Pacejka, *Tyre and vehicle dynamics*, 2nd edn. (Elsevier, Amsterdam, 2006), ISBN 0-7680-1702-5

[22] H.M. Choset, *Principles of robot motion: Theory, algorithms, and implementation*, Intelligent robotics and autonomous agents (MIT press, Cambridge, Mass, 2005), ISBN 0262033275

[23] M. Chitchian, A.S. van Amesfoort, A. Simonetto, T. Keviczky, H.J. Sips, *Adapting Particle Filter Algorithms to Many-Core Architectures*, in *IEEE 27th International Parallel and Distributed Processing Symposium (IPDPS), 2013* (2013), pp. 427–438, ISBN 978-1-4673-6066-1