DOI: 10.3217/978-3-85125-663-5-20

Traffic cone based self-localization on a 1:10 race car

Axel Brunnbauer[1] and Markus Bader[1]

***Abstract*— This document describes a feature-based self-localization running on-board on an automated 1:10 race car. Traffic cones are detected using an Intel RealSense depth camera and integrated into the self-localization as landmarks. The work presents a novel approach for how to detect traffic cones by fusing depth and RGB data. Motion commands and sensor readings are combined with an Extended Kalman Filter (EKF). The performance of the overall system was evaluated using an external motion capture system.**

I. INTRODUCTION

Learning the technical principles of autonomous driving with a real car is not only expensive but also dangerous. Therefore our work group uses multiple car models scaled at 1:10 with self-printed components for teaching, research and developing new approaches.
This work presents a localization system that is able to detect traffic cones in various contexts and, as a consequence, localize the vehicle using a prior known feature map. The car shown in Fig. 1 is an enhanced version of the model presented in [2]. The car is equipped with a depth camera and an Single Board Computer (SBC) running a high-level processing unit with Robot Operating System (ROS). The Brushless Direct Current (BLDC) motor for driving and the servomotor for the front Ackermann steering are controlled by a self-designed board connected by a serial cable to the SBC. A more detailed explanation of the model's components is provided later.
One challenging aspect of designing such a system is the detection of traffic cones. Difficulties in object detection are presented by various factors. Camera-based detection methods, in particular, are greatly influenced by changing lighting conditions. Currently, the car provides simple odometry by dead reckoning using Hall effect sensors on the motor and no wheel encoders. Therefore the number of revolutions of the motor and the state of the servomotors are used to calculate odometry. However, these measurements are subject to noise introduced by external and internal factors. External factors, on the one hand, constitute calibration errors regarding wheel diameter, wheelbase distance, slightly different wheel sizes, etc. Internal factors, on the other hand, are inaccuracies in the mathematical model used to describe the motion of the vehicle. Since the model is an abstraction of the real world, simplifying assumptions are necessary to keep the model feasible. The uncertainty presented by the errors in odometry and the sensor readings must be considered when building a reliable localization system. To cope with these issues, this work applies a variant of the Extended Kalman Filter (EKF) algorithm, which can deal with these types of uncertainties. Some other approaches dealing with localization and mapping of traffic cones are presented below, followed by a discussion of the methods applied in this project. Finally, the results of the experimental evaluation are compared with the current performance of the system.

*The research leading to these results has received funding from the Austrian Research Promotion Agency (FFG) according to grant agreement 855409 (AutonomousFleet) and 854865 (TransportBuddy)

[1]Axel Brunnbauer and Markus Bader are with the Faculty of Informatics, Computer Engineering, Vienna University of Technology, Vienna 1040, Austria, axel.brunnbauer@tuwien.ac.at

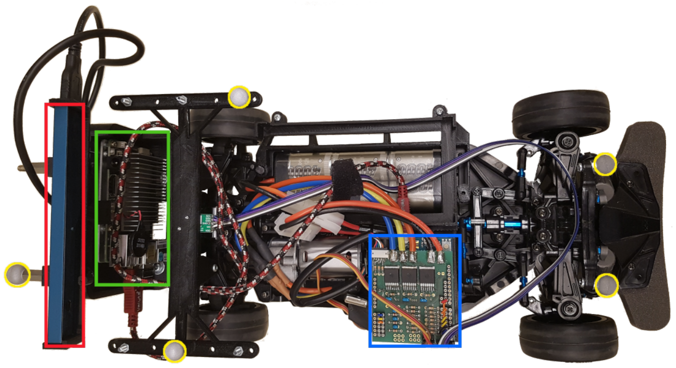


Fig. 1: 1:10 race with an on-board PC (green), a depth camera (red), a motor control board (blue) and reflectors (yellow) to record ground truth pose data with a motion capture system.

II. RELATED WORK

In [1], the development of the *flüela driverless*, the first car to win the Formula Student Driverless (FSD), was introduced. The *flüela driverless* is an electrically-powered race car equipped with a variety of sensors for traffic cone detection and state estimation, including a 3D LiDAR sensor, a stereo camera system, an inertial navigation system and an optical ground sensor. The vehicle is able to map the track defined by pairs of traffic cones using a particle-filter-based SLAM approach. Further, for localizing the car, a landmark-based EKF was implemented. The previously-mapped traffic cones serve as landmarks used for updating the current estimation of the vehicle's pose. To detect the traffic cones, the team used a 3D LiDAR scanner. Traffic cones are recognized by first removing the ground plane from the scan. Then, the filter clusters the traffic cones using the Euclidean clustering algorithm and further classifies these clusters according to size. In a second filtering step, traffic cones get rejected based on their distance from the sensor. The approach presented in this work also filters traffic cones based on their size. In addition, it rejects objects which do

not fit the visual appearance of the traffic cones.
Zeilinger et al. present their approach to designing an autonomous vehicle for the FSD in [5]. Their system uses a planar laser scanner and color cameras for the detection of traffic cones. Furthermore, to estimate the vehicle's movements, a rich set of sensors, including IMU's, GPS sensors, several wheel spin sensors and rotary encoders are mounted on the car. The traffic cone detection includes segmentation by color, exploiting the distinct color of the traffic cones. Also, techniques to incorporate the data from the stereo cameras and the laser scanner are presented. To map the traffic cones and localize the vehicle, an EKF-SLAM implementation was used. The work proposed in this paper also proposes an EKF implementation for localization, but does not take SLAM into consideration. The trajectory planning in [5] also exploits the fact that the pairs of cones already describe a path which have to be followed. This path is then smoothed using a model-predictive motion controller.

III. APPROACH

In the approach presented here, traffic cones are first detected and then located relative to the pose of the vehicle. Next, mapping is determined based on the perceived location of the landmarks and a known map.

A. *Traffic Cone Detection*

The algorithm developed for this task uses the guess and check method. First, objects in the vehicle's environment are located by extracting objects from the 3D point cloud. This is done by simulating a laser scan parallel to the ground. After potential candidates are detected, the algorithm filters out objects which do not correspond to distinct features of the landmarks. The first step of the landmark detection process is the extraction of objects from the laser scan. To extract objects from the set of points $P = \{p_1, p_2, \ldots, p_n\}$, the Euclidean clustering method proposed in [3] is used.
Filtering of the objects detected is realized as a filter pipeline where each filter takes the clusters extracted as an argument. In addition, each filter has individual input arguments necessary for detecting certain features of the objects. Currently, the pipeline consists of two filters. The first filter applied sorts out objects which do not correspond to the expected width of traffic cones, as estimated by the laser-based detection. Fig. 2 shows the laser scan obtained

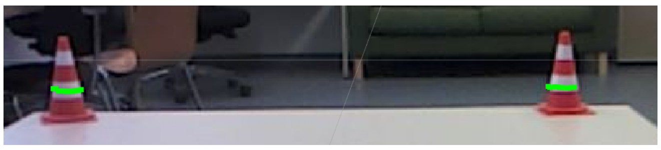
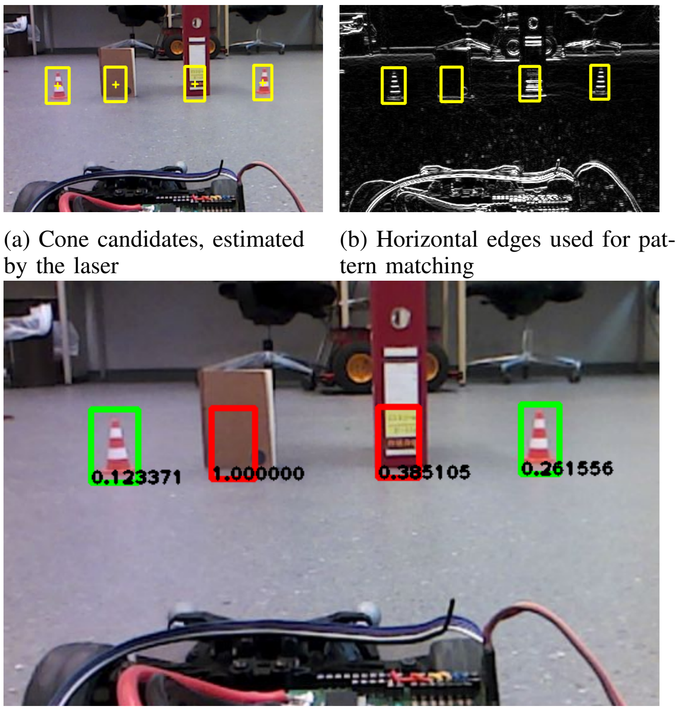


Fig. 2: Visualized laser scan readings in green on detected traffic cones

from traffic cones. If a segment does not correspond to a certain degree to the expected width, the segment is removed from the traffic cone candidate list.
The image-based filter exploits the visual properties of a traffic cone, especially its striped appearance. The classification of traffic cones is done using a template-matching algorithm. The goal of the algorithm is to match objects which correspond to the typical striped pattern of traffic cones. By applying the *Sobel* operator to find horizontal edges, unnecessary information is removed and the pattern of the traffic cones can be emphasized. Fig. 3b shows the extracted edges of the image. To speed up the template-matching process, the search space is reduced to a narrow area around the projected point. This area is defined by the traffic cone's height and width. First, the coordinates of the objects detected in the previous step are projected onto the image plane using extrinsic and intrinsic camera parameters. Now that the location of the candidate objects in pixel space is known, the search space is limited by the surrounding bounding boxes. In Fig. 3a, search spaces are marked with yellow bounding boxes. The marker inside a bounding box depicts the center of the detected object which is projected onto the image. The next step involves matching the sub-



(a) Cone candidates, estimated by the laser
(b) Horizontal edges used for pattern matching
(c) Final estimatation of traffic cones

Fig. 3: Image processing steps for traffic cone detection

regions of the image, defined by the candidate objects, with templates. For each bounding box in Fig. 3b, the template matching algorithm determines which template matches best. The method used for calculation is the *normalized sum of squared differences*, resulting in a value between 0 and 1. Values close to zero indicate strong similarities between a template and a candidate object. Then, a simple threshold operation is applied in order to reject regions with a matching score of higher than a certain value. Fig. 3c shows such matching using a threshold value of 0.3, which yielded good results throughout several experiments under various lighting conditions. The green bounding boxes indicate objects which have been matched, whereas the red ones indicate rejected candidate objects. A matching score is displayed at the bottom of each bounding box.

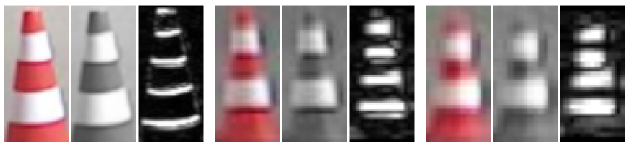


Fig. 4: Templates in different resolutions

Fig. 4 shows the templates used for template-matching in different resolutions and pre-processing steps. From left to right, each template is converted to a gray-scale image and then the horizontal edges are extracted. To get robust results, about forty templates were recorded from various angles and distances of up to 1.5 metres.

B. Localization

For the localization of the vehicle, a landmark-based EKF with unknown correspondences was implemented. Limited computational power and simplicity of implementation informed the selection of this approach. Implementation is based on the localization approach for unknown landmark correspondences, as proposed in [4]. Detected traffic cones are associated with the map using an Maximum Likelihood (ML) estimator which maximizes the probability of a detection being the landmark actually observed. To address the problem of false data association, a distance metric is used to reject landmark associations exceeding a predefined threshold. The metric of choice is the *Malahanobis* distance, as it is scale-invariant and unit-less. For a detected landmark $\boldsymbol{z}^i$ and landmark $\hat{\boldsymbol{z}}^j$ and its covariance $\boldsymbol{S}^j$, chosen by the ML estimator, the constraint shown in (1) must be satisfied.

$$\sqrt{(z^i - \hat{z}^j)^T [\boldsymbol{S}^j]^{-1} (z^i - \hat{z}^j)} < d_{max} \tag{1}$$

The EKF algorithm was implemented as closely as possible to the original. One challenge was embedding the algorithm into the ROS environment, since the original algorithm leaves out details on how to handle the complexity introduced by the dimension of time. In the real world, sensor readings and control updates are not received simultaneously or even at fixed intervals. The algorithm presented simply processes control and measurement updates in the order in which they arrive. That means that as soon as a measurement z arrives at time t, it is treated as it is perceived at the time it arrives. This causes the pose estimation to be biased in terms of it being always a few moments behind the actual pose. The reason for this is that when a measurement z is obtained at time t, the processing delay ε induced by the image detection is neglected. To separate the control updates from the measurement update, a global pose estimate is obtained via a series of relative transformations. Fig. 5 shows an example of such a transformation hierarchy.

The *map* frame depicts the initial pose at which the localization began. Control updates are successively incorporated into a transformation between a coordinate frame *odom* and

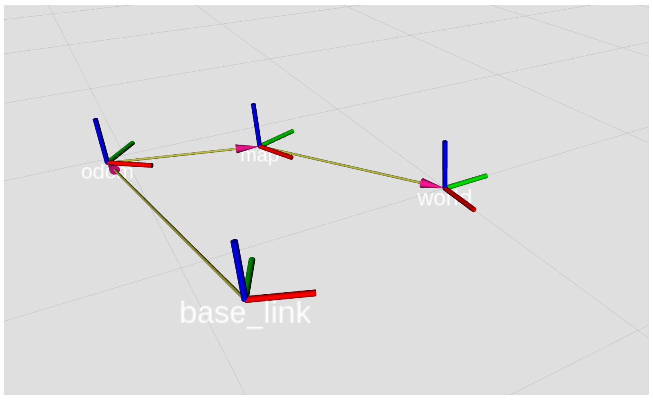


Fig. 5: Transformation hierarchy

a *base_link* frame. The transformation between these two frames corresponds to the position estimated by the odometry. Furthermore, a control update history is maintained. An example can be found in Fig. 6 in the *Odometry* queue. For each control update u_t, the most recent pose x_{t-1} relative to the *odom* frame's origin is selected from the queue. Then the control u_t is applied to x_{t-1} and Σ_{t-1}, respectively, and the result is appended to the queue. To determine the correction in this approach, another transformation between the *map* frame and the *odom* frame is applied. When a measurement z_t arrives at time $t + \varepsilon$, the last known pose before time t is queried from the queue. This pose is then transformed into the *map* coordinate frame and the EKF algorithm computes the correction. The transformation from the *map* frame to the *odom* frame can now be calculated by simply subtracting the transformation between *odom* and *base_link* from the correction computed. Since the approach

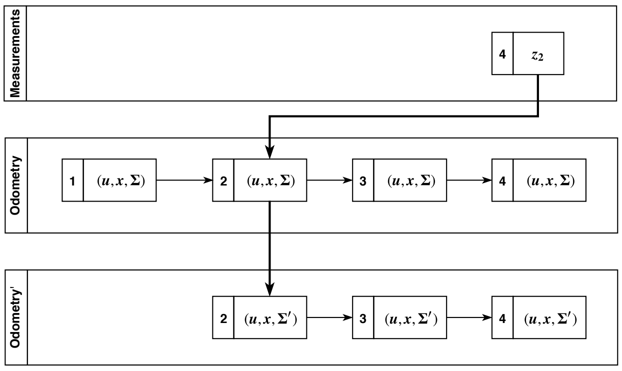


Fig. 6: Covariance update

so far only considers the vehicle's pose, another step is necessary to maintaining the covariance. The covariance of the queue entry at time t is replaced by the covariance of the corrected pose. Then, the motion model gets reapplied iteratively to all entries afterwards, while all entries prior to the one selected can be dismissed. Fig. 6 shows such a case. At time 4, a measurement obtained at time 2 arrives. The algorithm searches for an appropriate entry and makes the correction. Afterwards, the covariances are updated and all entries with a timestamp of under 2 are dismissed.

IV. EXPERIMENTAL RESULTS

To evaluate the approach presented so far, the proposed models are tested with a 1:10 scaled race car. The experiments were carried out in the case studies laboratory at the Institute. The laboratory is equipped with an OptiTrack[1] motion capture system allowing for the localization of visual markers with sub-millimeter accuracy. Fig. 7 shows the laboratory with the motion capture system indicated by red boxes. As shown in Fig. 1, the car is equipped with visual

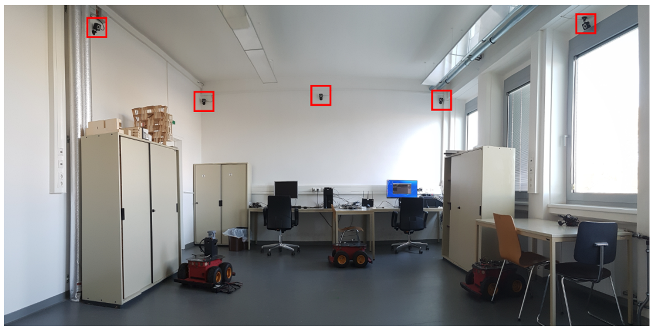


Fig. 7: Case studies laboratory, IR-cameras indicated in red.

markers which can be localized by the cameras.

A. *Traffic cone detection*

The evaluation of landmark detection accuracy was done by comparing the measurements with the true position of a traffic cone determined by the motion capture system. The dataset used contains about 4,000 measurements obtained from various distances and angles while driving the vehicle towards a single traffic cone with varying speeds. Fig. 8 shows the error of ρ in meters and the error of ϕ in radians. The data retrieved from the measurements suggests that uncertainty originates mainly from the distance measurement. The mean error of ρ, however, is approximately 4cm and therefore of systematic nature and can be compensated for easily. Below, Fig. 8 visualizes the distribution of the measurement errors. While the error of ϕ is almost distributed evenly around zero, the error of ρ is skewed towards the positive. The co-variance matrix used in the EKF algorithm was derived from this data, represented by the light blue covariance ellipse. Fig. 9a suggests that the error of ρ correlates with the distance to the traffic cone, while it is not significantly influenced by the angular offset to the vehicle. By applying polynomial fitting, a function describing the error of ρ regarding the distance to the traffic cone could be obtained. The function derived could be used to improve the localization algorithm by replacing the static sensor model by a dynamic approach.

B. *Localization*

To evaluate the accuracy of the localization algorithm, the vehicle was navigated through a number of test tracks. For each run, both the estimated pose, using only odometry, and the corrected pose, using traffic cone detection, were

[1]OptiTrack - Motion Capture System: https://optitrack.com/

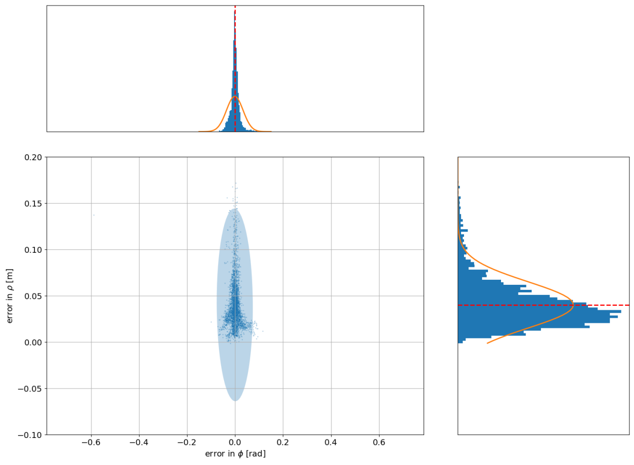


Fig. 8: Measurement error

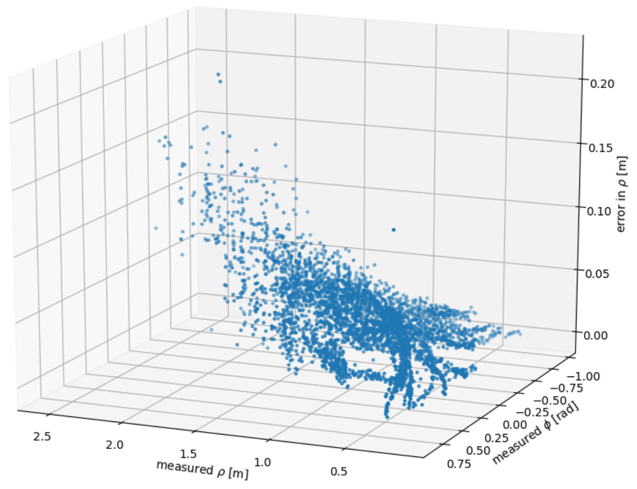
recorded simultaneously and compared.

Fig. 10 shows the performance of the self localization on a simple straight course. The blue points depict the ground truth position of the car obtained by the motion capture system. The orange points show the estimated position of the vehicle based on either odometry only (10a) or odometry combined with measurement updates (10b). The ellipses around each orange point show the covariance of the pose estimation and the blue crosses represent the traffic cones defining the course. A relatively strong deviation from the actual pose in Fig. 10a shows the inaccuracies of the mechanical configuration of the vehicle. One can see that the odometer always underestimates the actual speed of the vehicle. Furthermore, the vehicle tends to drift to the left. Therefore, some counter-steering was necessary to keep the vehicle on track when driving through the straight course.

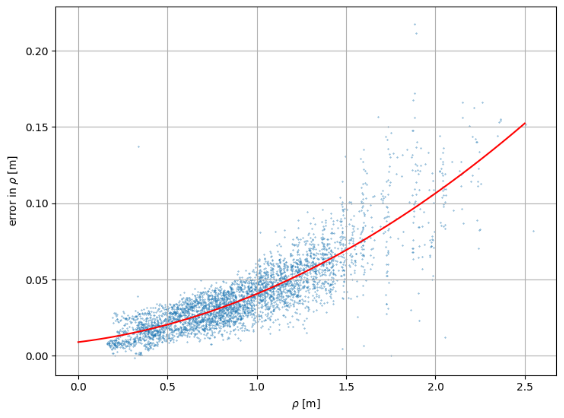
This is the cause for the odometry-based estimation in Fig. 10a drifting towards the negative y-quadrant. However, correction using traffic cone detection yields a significantly better estimation of the true pose, as can be seen in Fig. 10b. The deviation induced by drift could be compensated for almost completely. The evaluation reveals as well that the deviation from the true pose, interpreted as Euclidean distance, was able to be maintained at a relatively stable level when using the measurement update, while the approach using solely odometry shows that the vehicle tends to fall further behind the longer the run is.

Fig. 11 shows another test run through a slalom-like course. Again, the odometry base pose estimation shows great inaccuracies after a short period of time. Correction led to a more accurate estimation of the pose in this run, as well. In Fig. 11b, it can be seen how the vehicle's drift affects the estimation. After the second pair of traffic cones in a left turn, the deviation between actual position and estimation grows, due to the tendency of the vehicle to pull to the left.

Fig. 12 shows the evaluation of the localization on a circular course. The position estimation in Fig. 12a reveals, again, the sideways drift and the underestimation of the vehicle's speed.



(a) Error w.r.t location



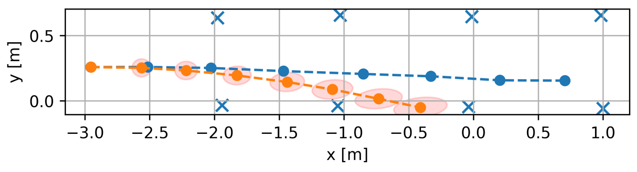
(b) Fitted error function of ρ

Fig. 9: Error with respect to different distances

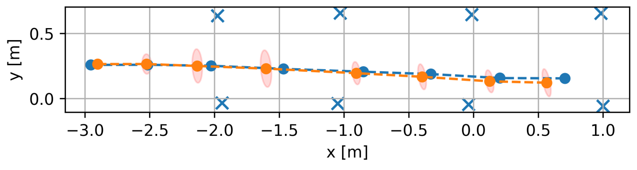
Whereas the car completed almost a whole lap on the course, the odometry-based estimation supposed that the vehicle completed only half of a lap. Although the correction does not work as well as on the previous courses, the estimation yields an improvement over the odometry estimation.

V. CONCLUSIONS

As the evaluation results suggest, the localization of the car can be improved by using the measurements obtained from the on-board camera. Although the odometry yields great uncertainty, pose correction can compensate for these inaccuracies in many instances.
An important step towards a more reliable self localization would be the improvement of the on-board odometry. Currently, only the speed of the wheels is estimated based on the revolutions of the motor and the wheel's diameter. For instance, a wheel encoder could be used to accurately determine the velocity of the vehicle. The localization system, as it is, requires that location of the traffic cones be known. The next step would be the implementation of a SLAM approach, to enable the vehicle to build up a map while driving a course. Potentially, in order to enable a vehicle

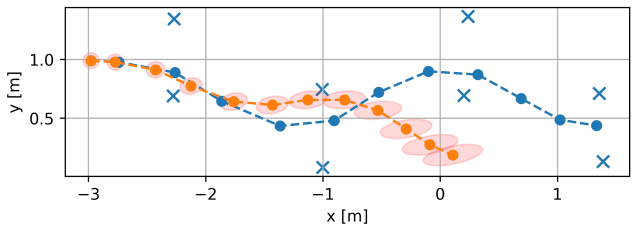


(a) Odometry only

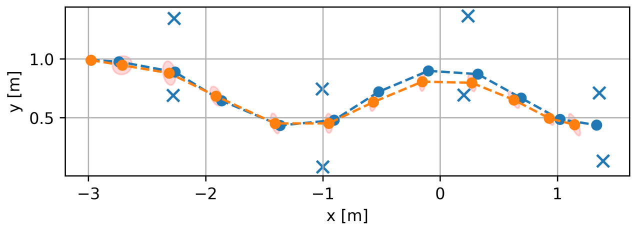


(b) Correction enabled

Fig. 10: *Straight course:* The estimated trajectory is shown in orange, the actual trajectory in blue. The figures show the position estimation of the vehicle based on its odometry, with EKF correction enabled.



(a) Odometry only



(b) Correction enabled

Fig. 11: Slalom course

to drive such a course autonomously, a navigation module could be developed that is able to navigate a path through a sequence of traffic cone pairs without colliding with any obstacles.

VI. APPENDIX

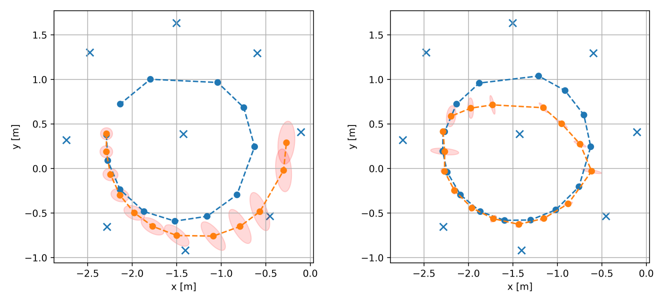
The data sets used to evaluate the components developed throughout this paper can be downloaded from the public repository[2].

REFERENCES

[1] M. de la Iglesia Valls, H. F. C. Hendrikx, V. Reijgwart, F. V. Meier, I. Sa, R. Dubé, A. R. Gawel, M. Bürki, and R. Siegwart, "Design of an autonomous racecar: Perception, state estimation and system integration," *CoRR*, vol. abs/1804.03252, 2018. [Online]. Available: http://arxiv.org/abs/1804.03252

[2] B. B. Eugen Kaltenegger and M. Bader, "Controlling and Tracking an Unmanned Ground Vehicle with Ackermanndrive," in *Proceedings of the Austrian Robotics Workshop (ARW-16)*, Wels, Austria, May 2016.

[2]https://github.com/axelbr/rccar-results



(a) Odometry only (b) Correction enabled

Fig. 12: Circular course

[3] R. B. Rusu, “Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments,” Ph.D. dissertation, Technische Universität München, 2009.

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

[5] M. Zeilinger, R. Hauk, M. Bader, and A. Hofmann, “Design of an Autonomous Race Car for the Formula Student Driverless (FSD),” in *Proceedings of the OAGM ARW Joint Workshop (OAGM ARW-17)*, W. K. A. M. B. B. Peter M. Roth, Markus Vincze and S. Stolc, Eds., Vienna, Austria, May 2017, pp. 57–62.