

Accurate Mapping and Planning for Autonomous Racing

Leiv Andresen^{1*}, Adrian Brandemuehl^{1*}, Alex Hönger^{1*}, Benson Kuan^{1*}, Niclas Vödisch^{1*}, Hermann Blum¹, Victor Reijgwart¹, Lukas Bernreiter¹, Lukas Schaupp¹, Jen Jen Chung¹, Mathias Bürki^{1,3}, Martin R. Oswald², Roland Siegwart¹, and Abel Gawel¹

Abstract—This paper presents the perception, mapping, and planning pipeline implemented on an autonomous race car. It was developed by the 2019 AMZ driverless team for the Formula Student Germany (FSG) 2019 driverless competition, where it won 1st place overall. The presented solution combines early fusion of camera and LiDAR data, a layered mapping approach, and a planning approach that uses Bayesian filtering to achieve high-speed driving on unknown race tracks while creating accurate maps. We benchmark the method against our team’s previous solution, which won FSG 2018, and show improved accuracy when driving at the same speeds. Furthermore, the new pipeline makes it possible to reliably raise the maximum driving speed in unknown environments from 3 m/s to 12 m/s while still mapping with an acceptable RMSE of 0.29 m.

I. INTRODUCTION

Autonomous racing has grown in popularity in the past years as a method of pushing the state-of-the-art for various autonomous robots. While competitions generally allow researchers to test the robustness and applicability of different solutions, racing in particular presents additional challenges to computational speed, power consumption, and sensing. Autonomous car racing started almost 16 years ago, with the DARPA Grand Challenge [1], and now continues with Roborace¹ and Formula Student Driverless (FSD)². In addition to car racing, new platforms such as quadcopters have recently begun to pick up momentum in new competitions like the IROS drone racing competition³.

Two main categories can be distinguished in autonomous racing: racing with and without prior knowledge of the track layout. For instance in Roborace, the race track layout is known a priori, whereas in IROS drone racing, the locations of a set of gates must be discovered during flight. FSD offers disciplines from both categories.

Racing on unknown tracks is challenging for SLAM algorithms. First, high-speed motion is more uncertain, which causes higher drift between sensor readings and poses challenges for data association. Furthermore, the horizon over which the upcoming dynamic motion must be planned scales with speed. If a tight corner is detected too late, the robot is unable to slow down in time for the turn, and inevitably

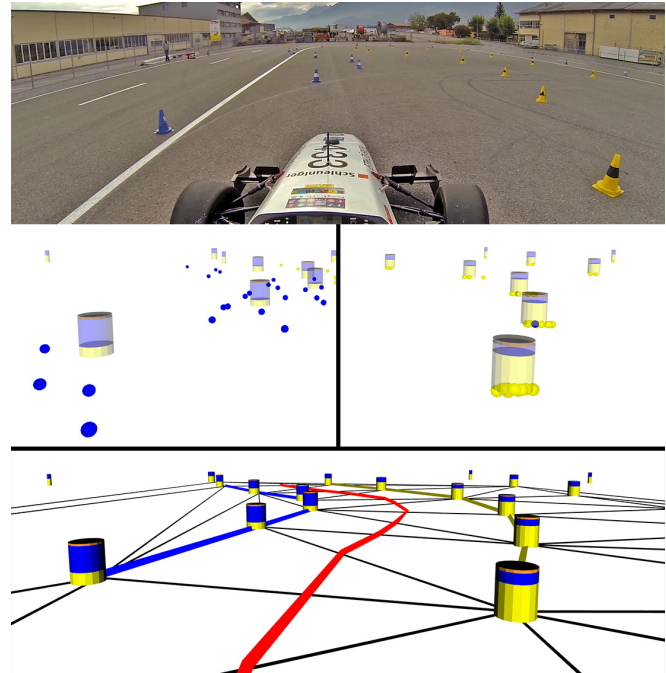


Fig. 1: Top: track in the front view of the car. Middle: accumulated observations of cameras (left), LiDARs (right), and filtered estimates (cylinders). Bottom: estimated track by boundaries and middle line.

leaves the track. This need for a longer horizon presents challenges not only for the planning algorithms but also for perception and mapping. For perception, the density of visual and 3D LiDAR sensor data is proportional to the inverse-square of the distance. Detecting distant landmarks is thus challenging and noise is unavoidable. The latter calls for a mapping pipeline that appropriately models and fuses the uncertainties of different modalities at varying distances. Finally, a planning approach is required that considers the uncertainty of the estimated map. In this work, we present a robotic system that is able to achieve high mapping accuracy while increasing the maximum speed on unknown tracks from 3 m/s [2], [3] to 12 m/s. Please refer to Fig. 1 and Fig. 2 for an illustration of the pipeline. A video demonstrating our work is available online⁴.

Our work is tested in the context of the FSD competition, where racing on both unknown and known tracks has been split after the 2018 season. The competition was previously

* These authors contributed equally to this work

¹ Autonomous Systems Lab, ETH Zürich. Contact: leiva@ethz.ch

² Computer Vision and Geometry Group, ETH Zürich

³ Sevenses Robotics AG

¹roborace.com

²www.formulastudent.de

³ris.skku.edu/home/iros2016racing.html

⁴https://youtu.be/1YBFcHY_gWE

conducted on an unknown ten lap course and autonomous racing pipelines [2], [3] focused on the reliability of a usually slow first ‘mapping lap’, knowing that nine fast laps would follow and reduce its effect on the total score. With racing in unknown environments as a separate one lap event (*Autocross*), the competition encourages systems to increase driving speed while still providing a high quality map for the consecutive ten lap event (*Trackdrive*) on the same course.

We present a perception, mapping, and planning pipeline designed to minimize lap time on an unknown track while still accurately mapping the track for the *Trackdrive* event.

The contributions of this paper are

- a perception pipeline that combines LiDAR and camera readings for accurate early fusion of cone detections,
- a two-stage mapping pipeline that allows for planning on lower certainty data while recording a high quality map,
- a new path planning algorithm utilizing Bayesian inference to find the track boundaries, and
- the integration of the aforementioned methods into a solution capable of planning over long horizons while driving at high speeds.

II. RELATED WORK

Most related to our work are the Roborace and the FSD competitions, both including races on known race tracks, where the emphasis lies on reliable localization and superior control. The authors in [4] and [5] deal with localization in known track environments, using only LiDAR and GNSS in combination with open-source ROS packages such as AMCL. Our previous works [2], [3], [6] also address this challenge by improving perceptual robustness with a novel detector and probabilistic sensor fusion scheme for cone observations on race tracks.

FSD also includes racing on a previously unknown race track, which became a separate discipline in its latest edition. Racing on an unknown track pushes requirements on detection range as it directly affects the achievable maximum speed. This is in addition to existing requirements on map quality as in previous AMZ works [2], [3], [6] and other FSD teams [7].

A key task of the presented perception system is the detection of cones. In [7], and [8] the authors present state-of-the-art CNN-based vision-only detection systems for cone detection based on versions of YOLO [9], [10]. In order to make better use of the accurate range measurements from LiDAR in combination with the rich semantic information from cameras, we implement an early sensor fusion approach for accurate color and position detection of cones. State-of-the-art object detectors that fuse camera and LiDAR data [11], [12] typically require powerful GPUs to run in real-time, making them unsuitable for racing systems where space and packaging for power supplies are challenging. We achieve real-time performance on a mobile version of the GTX 1070 without hitting the upper limit of our computing system.

We combine early fusion for cone detection with a Kalman Filter based local map. This yields improved cone localization and color estimation performance, by accumulating and filtering the measurements over time. For redundancy in case of sensor failures, vision-only and LiDAR-only measurements can still be incorporated in similar fashion to our previous work [2]. The authors in [13] develop a method for tracking dynamic obstacles, including discrete object categories, using late fusion via an Extended Kalman Filter. Similarly, [14] implements the fusion as an Unscented Kalman Filter (UKF). Since the landmarks that we wish to estimate are static, no nonlinear update terms are needed and we instead rely on plain Kalman Filters.

Once a partial map is observed, planning in partially known environments is well studied. Continuous state and action space planning methods such as RRT* and PRM* [15] are computationally complex, and do not exploit the specific patterns of the tracks that we aim to follow. Brandes et. al. [16] uses binary integer optimization on a fixed number of cones to find the immediate boundaries of a Formula Student track, however due to the low number of cones used it cannot give boundaries far enough from the vehicle to drive at high speeds safely. We develop an algorithm that is both computationally efficient and exploits the nature of the planned paths for highly accurate results with dynamic numbers of environment landmarks.

III. METHOD

A. Overview of the Autonomous Driving Pipeline

In FSD, tracks are demarcated by yellow, blue, and orange traffic cones (see Fig. 1). Yellow and blue cones mark the right and left boundaries, respectively, while orange cones mark the start line of the lap. In *Autocross*, track limits such as min/max cone separation, total track length, etc. are given by the rules [17], but the track is otherwise unknown.

Fig. 2 depicts the architecture of the autonomous driving pipeline we developed. The maximum speed of the car on an unknown track is directly limited by the distance at which the track boundaries can be reliably identified. Thus, the perception setup is designed to detect cones over a long range

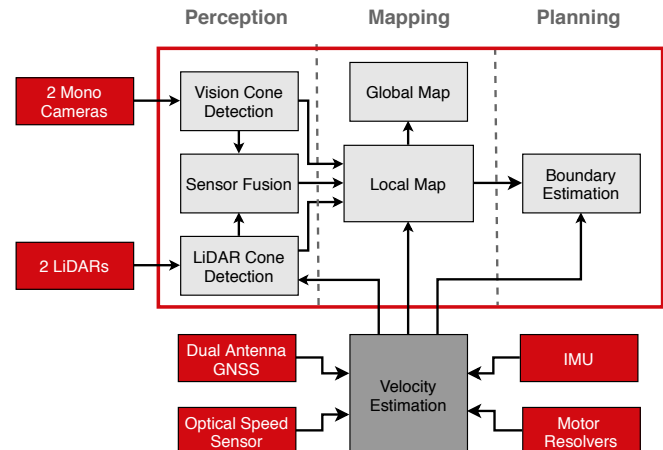


Fig. 2: Architecture of the autonomous driving pipeline.

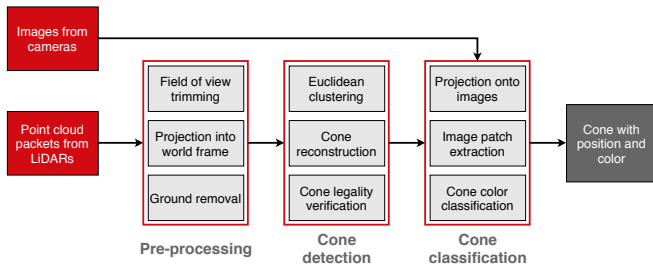


Fig. 3: The LiDAR pipeline used to detect cones from point clouds and to classify them according to their color via image patches. It consists of three major phases: pre-processing and cone detection following [2], cone classification is implemented by our early sensor fusion approach.

and with a large horizontal field of view. It comprises two 32 channel rotating LiDARs and two monocular cameras with global shutter; all devices are time synchronized to a common GNSS clock. To achieve better cone detection performance an early fusion approach that combines these two modalities was used. The longitudinal, lateral, and rotational velocities $\hat{v}^t = \{v_x, v_y, \psi\}$ of the car are estimated for ego-motion compensation and mapping. The estimate is obtained by fusing raw data from multiple sources including an Inertial Measurement Unit, dual antenna GNSS, an optical speed sensor, motor resolvers, and the motor currents with an Unscented Kalman Filter [18]. Utilizing the cones as landmarks, a local map fuses velocity estimates and the cone observations from the perception pipelines over time. The local map is used to estimate the track boundaries and middle path for the *Autocross* discipline with Delaunay triangulation and Bayesian inference. A globally consistent map is estimated using GraphSLAM to represent the cone positions accurately in the world frame for use in the later *Trackdrive* event. The definition of local and global mapping is further explained below in section III-C.

B. Perception

Position and color of cones can be detected independently by the camera and LiDAR pipelines. An optional early sensor fusion approach combines both modalities to improve the overall accuracy and robustness.

1) *Camera only*: The vision pipeline utilizes a convolutional neural network (CNN), Tiny YOLOv3 [10], that is trained to detect blue and yellow cones in the images of the mono cameras. Based on the size of the bounding boxes and the true dimension of a cone, the distance to the cone is estimated and the detected cones are projected into 3D space accordingly.

2) *LiDAR only*: The LiDAR pipeline consists of three stages: 1) pre-processing, 2) cone detection, and 3) cone classification. This pipeline follows the method described in [2], which presents an approach to estimate the color of a cone using only LiDAR point cloud data. A neural network predicts the most probable color based on the pattern of LiDAR intensity returns across multiple channels, e.g. a yellow cone is characterized by a bright-dark-bright pattern.

3) *Sensor Fusion*: Both the vision and LiDAR pipelines can run independently and therefore make the system robust to failures of single sensors, as was described in [2]. In the case that both sensor modalities are available, our sensor fusion approach on the input data level exploits the respective advantage of each modality. The overall pipeline is shown in Fig. 3. Once the positions of the cones have been determined using the aforementioned LiDAR-based cone detection, the points of each cone are projected onto the image of either camera depending on their lateral position. Since every point contains a timestamp, it is possible to interpolate them to the exact time of the image using the velocity estimates. The projected points of each cone are then framed by bounding boxes, as shown in Fig. 4. Since points on the lower part of a cone are often missing due to ground removal, the height of the initial boxes is multiplied by 1.5, which worked best in our experiments. Afterwards, the boxes' width is set to match their height. The resulting squares are scaled to a uniform size of 32×32 pixels, for subsequent classification.

Finally, the RGB patches are fed through a customized CNN, see Fig. 5, which is inspired by [2]. It consists of three convolutional layers, where each is combined with batch normalization, dropout, and a max-pooling layer, as well as four fully connected layers. Softmax is used to compute probability values for the three available classes: blue cone, yellow cone, and unknown color. To deal with occlusion, the training dataset contains samples of multiple cones of both mixed and the same colors within a single image patch, where the latter case is less critical for cone color detection.

C. Mapping

The three perception pipelines are filtered and fused together into a local map. In a second step, the output of this local map is used to build a global map of the whole track. The local and global map are detailed in the following:

1) *Local map*: Redundancy against sensor pipeline failures requires a mapping approach that works for all combina-

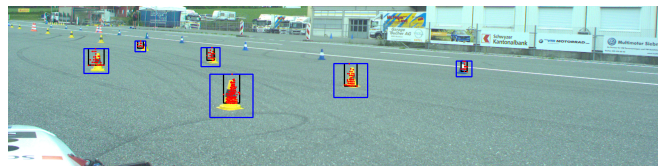


Fig. 4: Projection of the cones' point clouds onto both image planes. The points are shown in red, the initial bounding boxes are black, and the final image patches are indicated by blue squares.

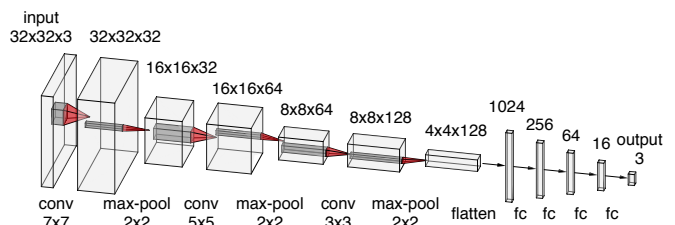


Fig. 5: CNN used to predict the cones' color based on RGB image patches.

tions of functioning pipeline subsets. Single sensor pipelines prove to be highly uncertain in either cone depth (monocular vision) or cone color estimation (LiDAR). Furthermore, all sensor pipelines suffer from significant noise when detecting distant cones. The global map requires observations to be sufficiently certain in order to avoid damage stemming from data association errors. On the other hand, path planning also requires knowledge of far and uncertain cones in order to evaluate sufficiently long planning horizons for fast driving. These two requirements can be satisfied by using a local map that estimates the position and color probabilities for all cones that were recently observed.

Sensor pipeline failures are detected by a time threshold and parameters are adjusted to one of four modes of operation. When the early sensor fusion pipeline is operational, no other pipelines are used to update the local map since they would fuse in redundant information. When the early fusion pipeline is not operational, the strengths of the individual modalities are modeled with low covariances on the cone positions from the LiDARs and cone colors from the cameras. Two modes of operation remain, which correspond to individual failures of the camera or LiDAR pipeline. Under pipeline failures, the goal shifts towards reliably finishing the lap, in order to still generate a map for the *Trackdrive* event. A speed of 5 m/s is tested and verified for all modes of operation.

A feature-based single hypothesis map $\mathcal{M}_{\text{local}}$ is chosen, since it is sufficient to accurately represent the cones that were recently observed. The pose estimate $\hat{x}^t = \{\mu_x, \mu_y, \mu_\theta\}$ of the car within $\mathcal{M}_{\text{local}}$ is computed by integration of the estimated velocity \hat{v}^t , starting at $\hat{x}^0 = \{0, 0, 0\}$. New cone observations \hat{z}^t are transformed from car frame to $\mathcal{M}_{\text{local}}$ using the synchronized sensor timestamps.

Cone positions and uncertainties are filtered using a Kalman Filter. The colors of the n observed cones at time t are represented by the random variable $\hat{C}^t = [\hat{C}_1^t, \dots, \hat{C}_n^t]$. The normalized sum of the probability distribution of \hat{C}^t over time yields a filtered color estimate up to time t . This filtered output is represented as $\tilde{C}^t = [\tilde{C}_1^t, \dots, \tilde{C}_n^t]$, where each cone i takes one of the following classes, $\tilde{C}_i^t = \{\text{yellow, blue, unknown}\}$. The motion uncertainty is incorporated by growing the pose covariance of all cones in $\mathcal{M}_{\text{local}}$ at each time step, using a hand tuned odometry covariance.

Associations of cone detections between time steps are challenging, as the noise levels on observed cone positions often exceed the minimal expected distance between neighboring cones. Each cone observation \hat{z} is therefore associated to its closest cone in $\mathcal{M}_{\text{local}}$ using the Bhattacharyya distance [19], which incorporates the position covariance of both the observation and the cone estimates in the map.

Furthermore, a negative observation model is used to reject non-existent cones resulting from false positives or outliers within less than 0.5s even for previously highly certain false positive cones. This is achieved by penalizing the certainty of cones in $\mathcal{M}_{\text{local}}$ that are not observed despite

being in the field of view of the sensors. The parameters of the observation model are derived from ground truth data analysis of the observation recall rates. The observation noise is also computed from the ground truth data.

An output $\mathcal{M}_{\text{local}}^{t_n}$ containing all cones within the map is created for every set of new observations measured at time t_n to be used by the global map and planning.

2) *Global map*: To drive multiple laps as fast as possible in the *Trackdrive* event, a global map containing all cones is generated and used for localization and racing line optimization. The map must be accurate to allow downstream algorithms to plan trajectories close to the cone delimited track boundaries.

The global map is generated in real-time on the car using a pose-landmark GraphSLAM algorithm [20]. Cones are used as landmarks, and the car and cone poses are jointly estimated via non-linear least squares using Google's Ceres solver [21].

Cone pose estimates are added to the (global) graph as virtual measurements from every new local map $\mathcal{M}_{\text{local}}^{t_n}$. In order to reduce the chance of false associations and outliers, new cones are only added to the global map if they were observed in the last perception reading and are in close proximity to the car. For as long as cones are in the local map, the data associations from $\mathcal{M}_{\text{local}}^{t_n}$ are re-used. Cones that are new in the local map are associated to landmarks in the graph using euclidean distance. If there is no landmark within a certain distance threshold, a new landmark is created. Graph edges connecting car poses are given by the integration of the velocity estimate between two subsequent local maps $\mathcal{M}_{\text{local}}^{t_n}, \mathcal{M}_{\text{local}}^{t_{n-1}}$.

Directly using the vision and LiDAR measurement models in the graph's factors would improve accuracy. However, we primarily use GraphSLAM to distribute the accumulated pose error when revisiting the finish line over the entire trajectory and map. Improving the measurement models and incorporating explicit place recognition will be the subject of future work.

D. Planning

In the *Autocross* discipline, the car must drive around an unknown track. Using the filtered cone observations from the local map as inputs, we estimate the track boundaries and middle path online in three steps. For the first and second steps, our approach follows the previous approach described in the Boundary Estimation section of [6]. In the first step, the search space is discretized using the Delaunay triangulation and the second step generates candidate middle paths from the triangulation. Our approach differs in the third step: it uses Bayesian inference instead of a single cost function to select the best middle path and its corresponding boundary points. The use of Bayesian inference allows the cone color probabilities to have a better representation of the validity of the path. This will be explained further in Section III-D.2.

The normalization term of the Bayesian inference is omitted as it does not affect the final selection result. Thus,

our inference equation consists of only three terms: the posterior $Pr(\text{valid path} | \tilde{\mathcal{C}}^t)$, the prior $Pr(\text{valid path})$, and the likelihood $Pr(\tilde{\mathcal{C}}^t | \text{valid path})$.

1) *Prior*: The prior is the probability of a cone configuration representing a valid path. The validity of a path is determined through human experience, which is captured in the form of six hand-crafted features. The function to calculate the prior value, consists of a prior weight W_{prior} and for each feature instance j , a feature value F_j , scaling factor N_j , setpoint value S_j , and feature weight W_j . It has the following form:

$$\text{Prior} = Pr(\text{valid path}) = e^{-W_{prior} \cdot \text{Cost}}, \quad (1)$$

$$\text{Cost} = \sum_{j=1}^6 \frac{W_j \cdot (F_j - S_j)^2}{N_j}. \quad (2)$$

The hand-crafted features, F_j , are defined as follows: F_1 for the largest angle change within the path; F_2 , F_3 and F_4 for the standard deviation of the distance between the left cones, the right cones and the track width respectively; F_5 for the number of triangulation edges crossed limited to the desired number; and F_6 for the length of the path.

2) *Likelihood*: The likelihood is the product of the color probability of the cones that are required to estimate the middle path from the prior. The color probability of a cone is distributed between the classes blue, yellow, and unknown. Cones on the left boundary can be either blue or unknown and will be assigned the higher probability value of either color. The same applies to cones on the right boundary which are either yellow or unknown. Cones that are not boundary points of the path will be assigned the highest probabilistic value amongst all three colors. The likelihood when n cones are detected, and the assigned color of each cone being represented by c_i , has the following form,

$$\begin{aligned} \text{Likelihood} &= Pr(\tilde{\mathcal{C}}^t | \text{valid path}) \\ &= \prod_{i=1}^n Pr(\tilde{\mathcal{C}}_i^t = c_i). \end{aligned} \quad (3)$$

With this probabilistic approach for the likelihood, the color probabilities of every observed cone are used to calculate the validity for each candidate path. As compared to the previous approach [6], where only the color probabilities of cones belonging to the boundaries of a candidate path are considered. This results in the color probability information of the other cones to be ignored.

3) *Posterior*: The aim of obtaining the posterior value for each possible middle path is for comparison, so that the middle path with the highest posterior value can be selected. To ensure numerical stability, log is applied to the new posterior. The formulation for the new posterior is now a sum of the log of the prior in section III-D.1 and the log of the likelihood in section III-D.2. This formulation is shown in the equation below,

$$\begin{aligned} \log(\text{Posterior}) &= \log(Pr(\text{valid path} | \tilde{\mathcal{C}}^t)) \\ &\propto \log(\text{Prior}) + \log(\text{Likelihood}) \\ &= -W_{prior} \cdot \text{Cost} + \sum_{i=1}^n \log(Pr(\tilde{\mathcal{C}}_i^t = c_i)). \end{aligned} \quad (4)$$

IV. IMPLEMENTATION

1) *Robotic platform*: The methods presented in this paper were implemented and tested on the autonomous race car *pilatus driverless*, a newer base vehicle compared to [2]. It is also powered by four electric motors and offers a full aerodynamics package.

All information is processed by a quad-core Intel Xeon E3-1505M with 16 GB of RAM using ROS and a real-time capable electronic control unit. The actuators include an electric power steering system, four self-developed wheel hub motors with regenerative braking, and an emergency brake system. Latencies for all pipeline elements can be found in Figure 6.

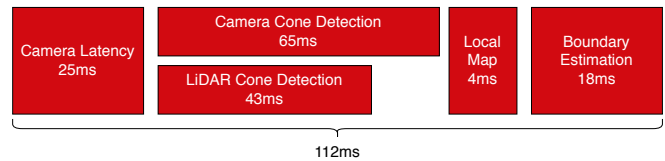


Fig. 6: Processing times of each pipeline component.

2) *RTK ground truth data*: Results in this paper are generated by comparison to ground truth data. Ground truth maps and trajectories are measured through GNSS with Real-Time Kinematic (RTK) corrections. A dual antenna GNSS RTK system installed on the car provides ground truth position and heading. A hand-held device with a second GNSS RTK system and user interface enables measuring cone locations and colors, and generates ground truth maps of the track with up to 2 cm accuracy for each cone. While usage of RTK is allowed in the competition, no guarantee of base station availability is made. Therefore, it is not used in the autonomous pipeline.

3) *Experimental setup*: Most experiments are conducted on *pilatus driverless*, the aforementioned race car, on various tracks at competitions and test sites. All tracks contain challenging segments, i.e. hairpins and parts of the track that come close to each other. The tracks' lengths vary between 200 m and 300 m and the race car's speed is predominantly in the range of 5-12 m/s. The results are compared to [2]. Where mentioned, some experiments use our approach on sensor data from [2] to account for the differences in vehicle and sensor setup.

V. RESULTS

A. Perception

We compared our novel low-level sensor fusion approach for cone color classification to the LiDAR-only method in

[2]. To account for the improved sensor setup, we re-trained this network with new data recordings.

Cameras capture richer color information than LiDAR and, hence, enable more robust cone classification. By artificially enlarging the area of interest around the projected points in the image, the method becomes more robust to imperfect projection due to errors in calibration and ego-motion estimation. Especially for larger distances, our image-based classification benefits from the rich representation. During the training of both networks, we only utilize data within a 10 m range. The results in Table I show that the performance of the intensity pattern-based method drops above 12.5 m, while the performance of our method does not change. Such an improvement enables higher certainty in the detection of the track and, ultimately, faster racing.

TABLE I: Accuracy of color classification of correctly detected cones with respect to distance.

Range [m]	0-5	5-7.5	7.5-10	10-12.5	12.5-15
previous approach [2]	0.88	0.93	0.89	0.87	0.80
our method	0.99	0.99	1.00	1.00	1.00

B. Mapping

We evaluate the accuracy and correctness of the global map generated by our mapping pipeline by aligning the estimated and ground truth maps with ICP [22], [23]. Then we compute the RMSE of the estimated cone locations with respect to their ground truth locations. We compare our results to the approach described in [2]. To compensate differences in the sensor setup, we applied our method to the same input sensor data used in [2]. Finally, we show the accuracy of our method on the improved sensor setup. The results are shown in Table II.

These results show that our method allows for more than a 4-fold increase in maximum mapping speed with a moderate decrease in mapping accuracy. Additionally, we show that our method achieves superior accuracy to the previous implementation when used on the same perception input data. Fig. 7 shows the mapping performance of our method for a 213 m long track recorded at a test site.

C. Planning

Using the parameter values, $W_k = 0.1 \mid k \in \mathbb{Z} : 1 \leq k \leq 5$, $W_6 = 0.5$ and $W_{prior} = 29$ for the Bayesian inference approach, we compare our implemented system against [6] using new data and recorded perception data from [2]. First, we evaluate our local map from section III-C.1 and

TABLE II: Comparison of mapping accuracy (RMSE) with various speeds and methods

Method	Dataset	Max. mapping speed	RMSE
previous approach [2]	2018 [2]	2.8 m/s	0.20 m
our method	2018 [2]	2.8 m/s	0.16 m
our method	2019	12 m/s	0.29 m

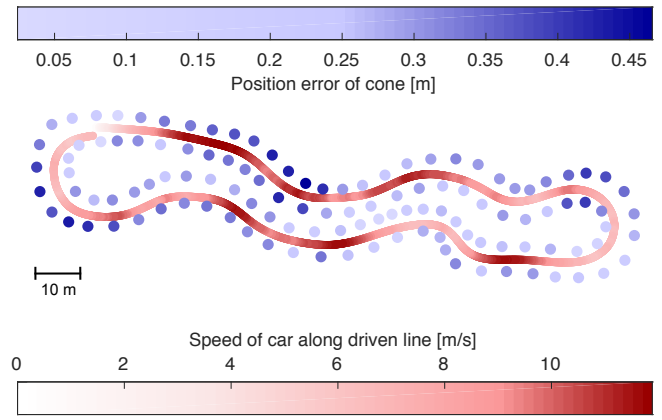
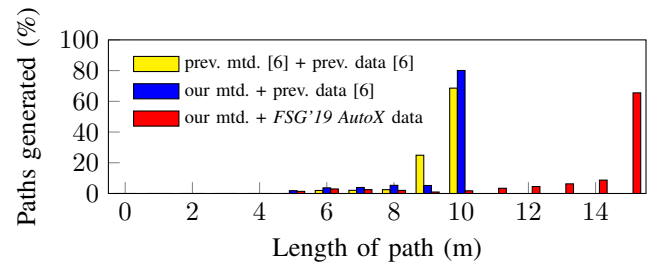
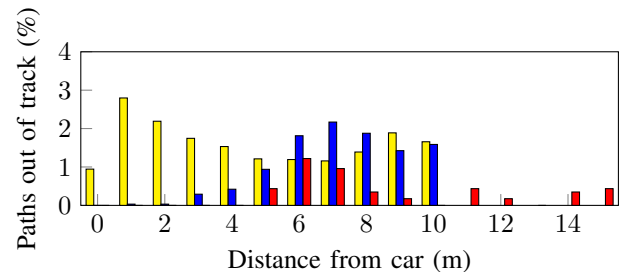


Fig. 7: Position error of the estimated cone locations relative to the ground truth. The driven line is colored according to the speed of the car.

boundary estimator from section III-D using the perception data from our previous work [6]. Second, we compare our entire pipeline using the data from our *Autocross* run in the FSG 2019 competition.



(a) Normalized histogram of percentage of tracks generated for different distance intervals from the car.



(b) Normalized histogram of percentage of estimated middle paths being outside of the actual track at a certain distance from the car.

Fig. 8: Histograms to assess the robustness of our system. The yellow and blue bars represent the previous and current implementations on the previous dataset [6], respectively. The red bar represents our implementation on *FSG 2019 Autocross* dataset.

Fig. 8a, verifies the robustness of our system by illustrating the distribution of the output paths according to the path length. It can be observed that when the previous perception data was used, paths were limited to a maximum length of 10 m, and most of the paths generated with this data had a length near the 10 m limit. Using the *FSG 2019 Autocross*

data, the maximum path length was increased to 15 m, due to the improvement in our perception range and accuracy.

Fig. 8b, shows that although most predicted paths reach the largest allowable length, some paths went out of the track midway. Generally, the shorter a path is before going out of the track, the lower the buffer for correction as well. Therefore, it is crucial for points on the path that are closer to the car to be estimated correctly. Furthermore, the histogram in Fig. 8b shows that the current implementation is more robust as the previous boundary estimation approach had a higher percentage of paths going out of the track at distances close to the car. Additionally, it can be observed that our improved perception is also a contributing factor, as the value of the red bar is either lower or about the same as compared to the other two bars for distances up to 10 m.

VI. CONCLUSION

This paper presented the perception, mapping and planning components of the autonomous driving system developed by the 2019 AMZ driverless team for the *Autocross* discipline. Our early fusion approach of combining point clouds and images achieved better results than a previous LiDAR-only approach across various distances. In mapping, a local map and global map were generated. Filtering the cone observations by fusing associated past and current cone observations generated the local map. Using only high-certainty observations from the local map, an accurate, globally consistent map was generated via pose-landmark GraphSLAM. Our experiments showed that compared to our previous implementation, a lower RMSE was achieved at the same mapping velocity, and the mapping velocity could be increased greatly without a large sacrifice on accuracy. Lastly, for planning, we presented a Bayesian inference method for estimating the middle path and its corresponding boundaries. We demonstrated higher robustness as a lower fraction of paths that left the track at distances nearer to the car when compared to our previous implementation on the same data set. These new methods enable faster driving with similar mapping performance, even at speeds 4 times the previous implementation.

ACKNOWLEDGMENT

The authors would like to thank the whole AMZ driverless team for their dedication and hard work in developing *pilatus driverless*, and the team's sponsors for their financial and technical support throughout the season. We also wish to express our appreciation for everyone at the Autonomous Systems Lab of ETH Zürich for their supervision and support throughout this project.

REFERENCES

- [1] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
- [2] N. Gosala, A. Bühler, M. Prajapat, C. Ehmke, M. Gupta, R. Sivanesan, A. Gawel, M. Pfeiffer, M. Bürki, I. Sa, R. Dubé, and R. Siegwart, "Redundant perception and state estimation for reliable autonomous racing," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 6561–6567.
- [3] M. I. Valls, H. F. C. Hendrixx, V. J. F. Reijgwart, F. V. Meier, I. Sa, R. Dubé, A. Gawel, M. Bürki, and R. Siegwart, "Design of an autonomous racecar: Perception, state estimation and system integration," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2048–2055.
- [4] A. Wischniewski, T. Stahl, J. Betz, and B. Lohmann, "Vehicle dynamics state estimation and localization for high performance race cars," in *IFAC-PapersOnLine*, vol. 52, July 2019, pp. 154–161.
- [5] T. Stahl, A. Wischniewski, 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, Jan. 2019.
- [6] J. Kabzan, M. d. l. I. Valls, V. Reijgwart, H. F. C. Hendrixx, C. Ehmke, M. Prajapat, A. Bühler, N. B. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, and R. Siegwart, "AMZ driverless: The full autonomous racing system," *CoRR*, vol. abs/1905.05150, 2019. [Online]. Available: <http://arxiv.org/abs/1905.05150>
- [7] K. Strobel, S. Zhu, R. Chang, and S. Koppula, "Accurate, low-latency visual perception for autonomous racing: Challenges, mechanisms, and practical solutions."
- [8] N. De Rita, A. Aymar, and T. Delbruck, "CNN-based object detection on low precision hardware: Racing car case study," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 647–652.
- [9] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.
- [10] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv e-prints*, p. arXiv:1804.02767, Apr. 2018.
- [11] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1–8.
- [12] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 918–927.
- [13] A. Ess, K. Schindler, B. Leibe, and L. V. Gool, "Object detection and tracking for autonomous navigation in dynamic environments," *The International Journal of Robotics Research*, vol. 29, no. 14, pp. 1707–1725, 2010. [Online]. Available: <https://doi.org/10.1177/0278364910365417>
- [14] X. Chen, X. Wang, and J. Xuan, "Tracking multiple moving objects using unscented kalman filtering techniques," 07 2012.
- [15] O. Arslan, K. Berntorp, and P. Tsiotras, "Sampling-based algorithms for optimal motion planning using closed-loop prediction," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4991–4996.
- [16] R. S. Kathleen Brandes, Allen Wang, "Robust lane detection with binary integer optimization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
- [17] Formula Student Germany, FS Rules 2019 v1.1.
- [18] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," *Signal Processing, Sensor Fusion, and Target Recognition VI*, vol. 3068, pp. 182–193, July 1997.
- [19] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distribution," *Bulletin of the Calcutta Mathematical Society*, vol. 35, pp. 99–110, 1943.
- [20] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Transactions on Intelligent Transportation Systems Magazine*, vol. 2, pp. 31–43, 12 2010.
- [21] S. Agarwal, K. Mierle, and Others, "Ceres solver," <http://ceres-solver.org>.
- [22] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992.
- [23] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," *Image Vision Comput.*, vol. 10, pp. 145–155, 01 1992.