2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Redundant Perception and State Estimation for Reliable Autonomous Racing

Nikhil Gosala*[1], Andreas Bühler*[1], Manish Prajapat*[1], Claas Ehmke*[1], Mehak Gupta*[2], Ramya Sivanesan*[2]
Abel Gawel[1], Mark Pfeiffer[1], Mathias Bürki[1,3], Inkyu Sa[1], Renaud Dubé[1,3], and Roland Siegwart[1]

***Abstract*— In autonomous racing, vehicles operate close to the limits of handling and a sensor failure can have critical consequences. To limit the impact of such failures, this paper presents the redundant perception and state estimation approaches developed for an autonomous race car. Redundancy in perception is achieved by estimating the color and position of the track delimiting objects using two sensor modalities independently. Specifically, learning-based approaches are used to generate color and pose estimates, from LiDAR and camera data respectively. The redundant perception inputs are fused by a particle filter based SLAM algorithm that operates in real-time. Velocity is estimated using slip dynamics, with reliability being ensured through a probabilistic failure detection algorithm. The sub-modules are extensively evaluated in real-world racing conditions using the autonomous race car *gotthard driverless*, achieving lateral accelerations up to 1.7G and a top speed of 90km/h.**

I. Introduction

Autonomous driving and its racing counterpart have received a lot of attention since the inception of the DARPA challenge in 2004 [1]. Fuelled by racing series like Roborace and Self Racing Cars, state-of-the-art algorithms have been developed to fulfill the requirements of real-world racing conditions [2, 3]. Despite major technological advances, developing reliable autonomous vehicles remains a challenge. For instance, in 2016, an autonomous vehicle failure was reported once every three hours in California alone [4]. To make autonomous vehicles safe and reliable, the robustness of both the sensor setup and the algorithms has to be enhanced. This paper aims to improve the reliability of the perception and state estimation pipelines by introducing algorithms that provide redundancy for processing data generated by multiple complementary sensors.

Several works in multi-modal perception focus on fusing measurements from different sensors to accurately estimate the robot state and map the environment. For example, multi-sensor fusion approaches are used to enhance object recognition and tracking [5, 6] or to find regions suitable for driving [7]. Although these approaches improve robustness and accuracy by fusing sensors, they do not provide redundancy in case of a sensor failure. For instance, if a visual sensor fails, the perception pipeline could lose either depth or semantic information, potentially reducing the robustness of the overall system. Sensor failure detection is also an active research area with steps being made towards outlier rejection [8], and detecting sensor malfunctions due to system attacks [9]. So far, redundancy has been achieved by replicating the sensor setup and running the pipelines using a voting based approach [10, 11]. However such approaches result in higher costs and computational demands.

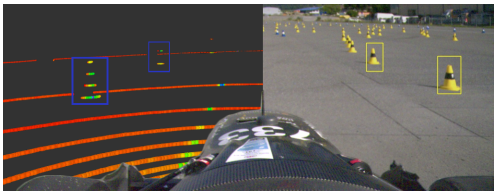


Fig. 1: The world as viewed by the event winning autonomous race car *gotthard driverless* using LiDAR (left) and cameras (right).

This paper presents a redundant architecture that enables reliable Simultaneous Localization and Mapping (SLAM) for an autonomous race car. The reliability of our perception module is improved by estimating the color and position of landmarks demarcating the track using both LiDAR and camera independently. Furthermore, the reliability of the velocity estimate is enhanced by the use of a failure detection module that can detect and isolate faulty sensors. These functionalities thus allow us to safely operate an autonomous race vehicle even under single-sensor failures. The approaches are experimentally evaluated with *gotthard driverless*, the autonomous race car that went on to win multiple Formula Student Driverless competitions in 2018.

The main contributions of this paper are (i) a learning-based approach to estimate landmark colors using LiDAR measurements, (ii) an EKF-based slip aware velocity estimator with probabilistic failure detection, and (iii) a particle filter based SLAM algorithm for fusing multi-modal landmark observations.

II. Problem Statement

The objective of this work is to enable a race car to autonomously complete multiple laps of an unknown race track ($\sim$500 m long) without any human intervention and in a single attempt. The left and right boundaries of the race track are assumed to be demarcated using blue and yellow cones respectively [12]. The two main challenges faced in such scenarios are (i) the lack of prior knowledge of the race track, and (ii) the possibility of a sensor failure hindering the operation of the car.

To enable autonomous navigation, the race car is equipped with a 3D LiDAR, and three color cameras in a mono and stereo setup. An inertial navigation system (INS), an optical ground speed sensor (GSS), and four wheel speed sensors (WSS) allow for real-time state estimation.

III. Method

This section describes the approaches developed for the reliable operation of an autonomous race car. To ensure that

* The authors contributed equally to this work.
[1] Authors are with the Autonomous Systems Lab, ETH Zürich, Zürich.
[2] Authors are with the CVG Group, ETH Zürich, Zürich.
[3] Authors are affiliated with Sevensense Robotics Ltd, Zürich.

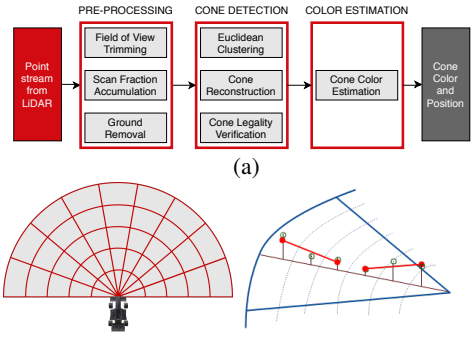


Fig. 2: (a) The LiDAR pipeline for cone color and position estimation in real-time. (b) Top view of the segmentation of the ground into sectors and bins. (c) Isometric view of the adaptive ground removal in one sector. Red lines represent the fitted ground lines [13].

the car stays within the track limits, knowledge of the cones' position and color is required. The observations gathered by the perception sensors are fused with the velocity estimate to guarantee reliable mapping and state propagation, thus ensuring successful navigation around the race track.

A. LiDAR Cone Detection and Color Estimation

The first step towards redundancy in perception is achieved by estimating the color and position of cones using LiDAR only. The sub-system architecture is depicted in Figure 2a, the main elements of which are described below.

1) Pre-Processing: Motion distortion in LiDAR scans is compensated using velocity estimates, after which distortions as large as $2\,\mathrm{m}$ in a single scan are reduced to only $2.6\,\mathrm{cm}$. The ground points from the resulting point cloud are removed using an adaptive ground removal algorithm [13] that adapts to changes in inclination of the ground. The ground is split into multiple sectors and bins (Figure 2b) and lines are fit through the lowest points of each bin (Figure 2c). Finally, all points within a threshold of the closest line are removed.

2) Cone Reconstruction and Filtration: Ground removal also results in the removal of nearly 64% of cone points on average, which reduces the number of points per cone and makes cone identification challenging. This is overcome by clustering the points after ground removal using the euclidean distance based approach [14, 15], and reconstructing a small cylindrical area around each cluster using points from the distortion-free point cloud. The reconstructed clusters are passed through a filter that checks whether the number of points in the cluster matches the expected number of points in a cone at that distance, which is computed using (1):

$$E(n_d) = \frac{1}{2} \times \frac{h_c}{2 * d * \tan(\frac{r_v}{2})} \times \frac{w_c}{2 * d * \tan(\frac{r_h}{2})} \qquad (1)$$

where n_d is the number of points at distance d, h_c and w_c are the height and width of the cone respectively, and r_v and r_h are the vertical and horizontal angular resolutions of the LiDAR respectively. The clusters that pass through the filter are then propagated to the color estimation module.

3) Color Estimation: The color estimation uses the repeatable intensity patterns in the point cloud obtained from the cones. Figure 3a shows the cones and the varying intensity order as one moves along the vertical axis of the

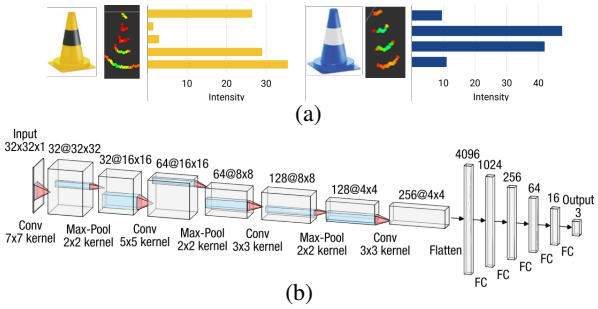


Fig. 3: (a) Intensity gradient for yellow and blue cone. (b) CNN used for color estimation.

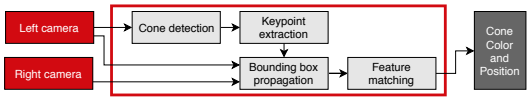


Fig. 4: Vision system architecture with images as the input, and color and 3D position estimates of the cones as output.

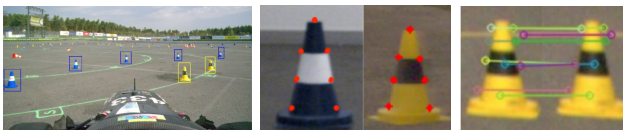
cones. This differing intensity order is capitalized upon, and the color is estimated using a Convolutional Neural Network (CNN) (see Figure 3b). To improve the generalization of the network, dropout and batch normalization are used. Additionally, incorrect predictions are penalized using an asymmetric cross-entropy loss function that penalizes misclassifications by a factor of 100. The CNN accepts a 32x32 grayscale image of a cone with pixel values representing intensities of points in the point cloud, and outputs the probability of each cone being *blue*, *yellow*, and *unknown*. The input image is created by mapping the 3D bounding box of the cone cluster to a 32x32 image where the cluster center is mapped to the image center and all the other points are appropriately scaled to fit in the image. The intensities of the mapped points are then scaled by a constant factor to make the disparity between different layers more apparent.

We hypothesize that compared to a rule-based classification approach, the CNN offers higher robustness to noise and is capable of learning hidden patterns from the input data. Furthermore, the color estimation is capped at $5\,\mathrm{m}$, because the sparsity of the point cloud above this distance does not allow for a distinction between the color patterns.

B. Visual Cone Detection and Stereo Pose Estimation

The cones' colors and positions are estimated by the stereo camera in addition to the LiDAR. The presence of multiple identical cones in an image poses a challenge for matching corresponding cones across images. This is overcome by detecting cones using YOLOv2 [16] in only one image, and spatially propagating the bounding boxes to the other by exploiting the prior knowledge of their appearance. The 3D position estimate is then improved by triangulating only the specific patches of interest instead of the complete stereo image pair. The major components of the pipeline are described below and its architecture is shown in Figure 4.

1) Cone Detection: YOLOv2, which offers good accuracy while being computationally efficient is used to detect cones in the images (Figure 5a). It is trained on two classes: *blue* and *yellow* cones. The network parameters like the anchor box size, non-maximum suppression and confidence thresholds are tuned on a self-acquired dataset to reduce both



(a) (b) (c)

Fig. 5: (a) Bounding boxes from YOLOv2 on image from left camera. (b) Keypoints regressed on cones (c) Feature matching in corresponding bounding boxes.

false positives and inaccurate bounding boxes. The network is trained on images with varying illumination and weather conditions for better generalization and robustness in real-world applications.

2) Keypoint Extraction: As the shape and size of the cones demarcating the track are standardized and fixed, a representative 3D model of a cone is generated and is used along with the corresponding 2D keypoints to calculate the pose of the cone with respect to the camera via the perspective-n-point algorithm (PnP) [17]. To extract these 2D keypoints in a bounding box, a neural network[1] is developed [18]. Inspired by classical computer vision, where corners are among the most prominent features in images, this neural network regresses on 7 such keypoints (Figure 5b) using cross-ratio in its loss function. This results in 3D position estimates of cones in the left camera's coordinate system.

3) Spatial Bounding Box Propagation: The 3D position estimates of the cones obtained from the left camera's image are expressed in the right camera's coordinate system using the stereo camera calibration. The cones are then projected onto the right camera's image plane. Using stereo geometry, the bounding boxes from the left camera's image are thus spatially propagated to the right camera's image. The precision of the propagation is enhanced by introducing constraints based on disparity and epipolar geometry.

4) Feature Matching and Triangulation: The final step in obtaining the improved 3D position estimate is to triangulate the corresponding pair of left and right bounding boxes. SIFT features [19] (robust to scale, rotation and illumination) are extracted and matched across the bounding box pair using brute-force matching (Figure 5c). The matched features are triangulated and the median of these triangulated points is then used as the 3D position estimate of the cone. Median is preferred to mean because the latter is susceptible to outlier matches which would result in incorrect 3D position estimates.

C. Velocity Estimation

Robust and accurate velocity estimates are key for reliable operation of autonomous cars. They are used to propagate poses, compensate for motion distortion in LiDAR data, and influence accuracy of SLAM and efficiency of control algorithms. Compared to typical mobile ground robots, race cars have high wheel slip, up to 20% for optimal longitudinal acceleration [20], which strongly biases wheel odometry. In addition, velocity sensors like GNSS and GSS are prone to failure or bias in rough environments (e.g. cloudy, cluttered environments [21], or wet surfaces [22]). These challenges are addressed by estimating the slip within a probabilistic framework.

[1]The neural network architecture is part of a separate contribution.

There exist a variety of filtering [23] and batch optimization approaches for state estimation [24]. EKF was selected due to its computational efficiency, ease of debugging, access to state covariances and due to its proven performance in real-world applications [8].

This section presents the process and measurement models followed by a failure detection module that isolates the posterior from false likelihood measurements. A simplified architecture is shown in Figure 6.

1) Process Model: The car is assumed to remain in contact with a flat surface, allowing the model to be built in 2D. The state vector $\mathbf{x} \in \mathbb{R}^{9\times1}$ is defined as

$$\mathbf{x} = [\mathbf{v},\ r,\ \mathbf{a},\ \mathbf{sr}]^T, \tag{2}$$

where $\mathbf{v} = [v_{\text{x}}, v_{\text{y}}]$ and r represent the linear and angular velocities respectively, $\mathbf{a} = [a_{\text{x}}, a_{\text{y}}]$ denotes the linear accelerations (note that we omit the vertical component of velocity and acceleration), and $\mathbf{sr}_{\text{ij}} = [sr_{\text{FL}}, sr_{\text{FR}}, sr_{\text{RL}}, sr_{\text{RR}}]$ denotes the slip ratio of wheel ij, where i $\in$ {Front, Rear} and j $\in$ {Left, Right}, defined as:

$$sr_{\text{ij}} = \begin{cases} \frac{\omega_{\text{ij}} \cdot R - V_{\text{ij}}}{V_{\text{ij}}} & \text{if } |V_{ij}| > 0 \\ 0 & \text{if } |V_{\text{ij}}| = 0, \end{cases} \tag{3}$$

where ω_{ij} and V_{ij} are the angular and linear velocities of wheel ij and R is the radius of the wheel. The process model represents a prior distribution over the state vector wherein the velocity is propagated using a constant acceleration model, and slip ratios are propagated using the dynamics derived by time differentiation of slip ratio given by 3. The process model is defined as:

$$\begin{aligned} \dot{\mathbf{v}} &= \mathbf{a} + [v_{\text{y}} r,\ -v_{\text{x}} r]^T + \mathbf{n}_{\text{v}} \\ \dot{r} &= f_{\text{M}}(\mathbf{sr}, \mathbf{v}, r, \delta) + n_{\text{r}} \\ \dot{\mathbf{a}} &= \mathbf{n}_{\text{a}} \\ \dot{\mathbf{sr}} &= \frac{\mathbf{T}_{\text{M}} \cdot R}{\mathbf{V}_{\omega\mathbf{x}} \cdot I_\omega} + \frac{\mathbf{sr}}{\mathbf{V}_{\omega\mathbf{x}}} \cdot \left(\frac{C_\sigma \cdot R}{I_\omega} - a_{\text{x}}\right) - \frac{a_{\text{x}}}{\mathbf{V}_{\omega\mathbf{x}}} + \mathbf{n}_{\text{sr}} \end{aligned} \tag{4}$$

where $f_{\text{M}}(\cdot)$ computes yaw moment based on tire forces estimated using a linear function of longitudinal and lateral slip (see Chapter I [25]). $\frac{(\cdot)}{\mathbf{V}_{\omega\mathbf{x}}}$ denotes element-wise hadamard division. Motor torque $\mathbf{T}_{\text{M}} \in \mathbb{R}^{4\times1}$ and steering angle δ are input to the process model and measured using a current sensor and an encoder respectively. C_σ is the longitudinal tire stiffness [25], $\mathbf{V}_{\omega\mathbf{x}} \in \mathbb{R}^{4\times1}$ is the longitudinal velocity of the wheel hub, I_ω is the moment of inertia of the wheel, and $\mathbf{n}_{\{\cdot\}}$ is the i.i.d. Gaussian white noise. These terms enable the process model to capture the fact that the probability of slippage increases with increase in motor torque.

2) Measurement Model: The measurements from all the sensors can be combined to update one or more of the following state variables:

Slip Ratio: Slip ratios are updated using the WSS measurements. Observability analysis [8] concludes that with a certain combination of faulty sensors, the state variable becomes unobservable. To predict velocities in such cases, the model is switched from a full dynamic model to its partial kinematic counterpart by updating the slip ratios with a *zero*

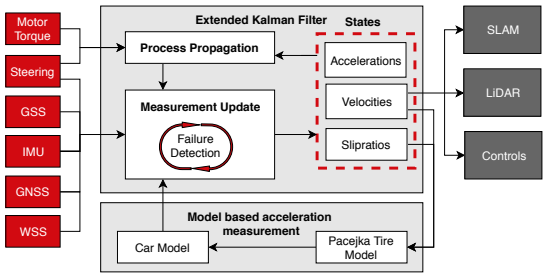


Fig. 6: A simplified velocity estimation architecture

slip ratio update (ZSU). High changes in wheel speeds are captured as slip by the process model and even though the ZSU later shrinks the slip ratio in the update step of the EKF, the velocity estimate remains reliable.

Acceleration: Accelerations are estimated using the IMU and the car's dynamic model. The IMU is considered reliable, yet on inclined surfaces the 2D assumption is violated and the gravity vector corrupts the true lateral and longitudinal accelerations. On the other hand, the dynamic model for calculating acceleration is based on the Pacejka model [20], which is accurate for small slip but is influenced by environmental conditions. Robustness is increased by fusing data from both these sources which is made possible by the inclusion of accelerations in the state vector.

Velocity: Linear velocities are updated using the GSS, GNSS and WSS whereas the angular velocity is observed using the IMU in addition to the above three.

The measurement model $\mathbf{z} \in \mathbb{R}^{13\times 1}$ is given by:

$$\begin{aligned} \mathbf{z}_{\mathrm{v}} &= h_{\mathrm{v}}(\mathbf{x}) = \mathbf{R}(\theta_{\mathrm{s}})(\mathbf{v} + [-r\,\mathbf{p}_{\mathrm{s,\,y}},\; r\,\mathbf{p}_{\mathrm{s,\,x}}]^T) + \mathbf{n}_{\mathbf{z_v}} \\ z_r &= h_r(\mathbf{x}) = r + n_{z_r} \\ \mathbf{z_a} &= h_a(\mathbf{x}) = \mathbf{a} + \mathbf{n}_{\mathbf{z_a}} \\ \mathbf{z}_{\omega} &= h_{\omega}(\mathbf{x}) = \mathbf{V}_{\omega\mathbf{x}} \cdot (\mathbf{sr}+1)/R + \mathbf{n}_{\mathbf{z}_{\omega}} \end{aligned} \tag{5}$$

Here, $\mathbf{R}(\theta_{\mathrm{s}})$ denotes the rotation matrix where θ_{s} is the orientation of the sensor in the car frame and $n_{\{\cdot\}}$ is the i.i.d. Gaussian noise that corrupts the sensor measurements.

3) Sensor Failure Detection: Since inaccurate sensor measurements (e.g. sensor failure) can result in poor state estimation for the next iteration, recognition of such abnormal sensor status is critical for the recursive algorithm. The sensor faults can be classified as *outlier* (e.g. spikes in measurements), *drift* and *null*. A Chi-square-based approach similar to [8] for outlier detection, and a variance based sensor isolation for drift detection given by 6 are implemented.

$$\sum_{i=1}^{n} (\mathbf{z}_{\mathrm{i}} - \mu_z)^2 < k \tag{6}$$

μ_z represents the mean of the sensor measurement. For each measurement, the variance is calculated with $n(>2)$ number of sensors used to measure that state variable. If the variance exceeds a tunable parameter k, sensor readings are discarded progressively until the sensor with the highest contribution to the variance is rejected at the given time instance.

D. Localization and Mapping

To unfold the car's full dynamic potential, a planning horizon with at least 2 s look-ahead is required. At high speeds it is infeasible to perceive upcoming corners for such a long horizon ($\sim$30 m). Therefore, a map is built at low speeds and afterwards used to localize the car and plan maneuvers. Since the landmarks (LMs) are of similar appearance and can only be distinguished by their color and position, the algorithm should be able to incorporate probabilistic LM identification. It must additionally run in real-time and its runtime should be easily tunable to allow for adjustments if necessary. The fastSLAM 2.0 [26] algorithm is selected as the particle filter structure inherently allows for computationally efficient independent data associations, compared to multi-hypothesis approaches for EKF-based SLAM [27] or graph-based methods. Since it does not utilize an optimization step its runtime is lower and more predictable [28], and its accuracy is higher than that of its predecessor (see Figure 13 in section IV-D). Figure 7 illustrates the SLAM architecture. The LiDAR and camera pipelines are treated independently, providing observations at different frequencies, with different delays, and different uncertainty models. The output of the mapping and localization pipeline is a 2D feature-based map, and a pose within this map, both compensated for delay.

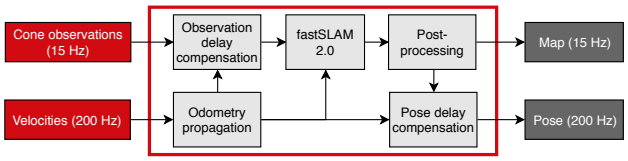


Fig. 7: SLAM Architecture during mapping phase with observations and velocity estimate as inputs and a resulting map and pose estimate as output.

1) Mapping phase: The map is updated every time new LM observations are received. Each observation consists of a position estimate $\mathbf{z}_{k-\Delta k}$ in the local vehicle frame at time step $k-\Delta k$ and a color estimate $\mathbf{c}_{obs,k} = \mathbf{c}_{obs,k-\Delta k} = [p_{blue}, p_{yellow}, p_{other}]^T$. The delay is compensated by propagating the observations forward using the motion estimate between time step $k-\Delta k$ and k. To update the filters for each particle, a new particle pose is predicted through an odometry motion model without noise [28]. The LM observations are then associated with the already existing LMs in the map using the nearest neighbour method [29], where the Mahalanobis distance is used as a measure for the likelihood of correlation and to find the data association $\mathbf{a}_k$ for each particle. If the maximum likelihood of an observation correlating to any LM is below a threshold l, the observation is assumed to belong to a new LM. For every particle an EKF is used to enhance the pose accuracy. The prior is given by the odometry propagation, and the posterior is iteratively refined by incorporating the matched LMs. After drawing a pose from the generated distribution, the position estimates of the observed LMs are updated in a straight forward manner using standard EKF equations.

2) Color Integration: The LM colors are modelled as a categorical distribution with $K = 3$ possible outcomes. A color estimate is drawn from the distribution provided by each sensor. The respective counter α_i is then increased. In the color probability distribution of each LM, the probabilities are set to their expectation value [30]:

$$p_i = \mathbb{E}[p_i] = \frac{\alpha_i}{\sum_{k=1}^{K} \alpha_k} \tag{7}$$

Depending on the number of measurements received and the type of sensor, a color is drawn from the respective distribution using the maximum-a-posteriori method.

3) Particle Weighting: Each particle is weighted according to how well the observations match the already existing map. The total weight w_k of a particle at time step k is given by:

$$w_k = w_{k-1} * l^{\nu} * w_b^{\kappa} * w_c^{\gamma} * \prod_{n \in \mathbf{a}_k} w_{k,n} \tag{8}$$

where ν is the number of new LMs, κ the number of LMs that are in sensor range but were not observed (penalized by w_b per LM), γ the number of LMs whose color did not match (penalized by w_c per LM) and $w_{k,n}$ the weights of all matched LMs, which are computed after updating the EKF for each LM. The weights of all particles are then normalized. The particle weight variance naturally increases over time and therefore resampling is enforced once the effective sample size N_{eff} over the total number of particles drops below a given threshold.

4) Failure Detection: For each new set S of LM observations, a sensor failure detection step is applied after data association and used to reduce map quality degradation due to irreversible EKF updates. The observation ratio of a LM is defined as the number of times the LM has been detected over the total number of times it was in a sensor's perceptual field of view (FoV). The set S is accepted only if enough observations match with landmarks that have an observation ratio above 80%, given there are any in the FoV.

5) Post-Processing: After one driven lap, a track loop is detected when all particles collapse within $4\,\mathrm{m}$ around the start with a standard deviation of less than $0.2\,\mathrm{m}$ and a similar orientation compared to the beginning of the race. Subsequently, the boundaries of the track are estimated using the map of the highest-weighted particle. The LMs are classified as inside and outside and ordered according to the previously driven line.

6) Localization: After the first lap - in case of loop closure - the EKF map update of the fastSLAM 2.0 algorithm is disabled, which essentially turns it into Monte Carlo localization. The pose estimate is then computed as the mean of all particle poses.

IV. Results

The approaches proposed in this paper are deployed on the autonomous race car *gotthard driverless*. It is based on *gotthard*, an electric four-wheel drive race car with a full aerodynamics package built by AMZ[2] in 2016. Additional sensors and actuators were added to enable fully autonomous operation. The car was tested in five different testing locations in addition to the competitions, and all the data presented in this section has been gathered during these runs.

A. LiDAR Cone and Color Detection

Figures 8a and 8b compare the performance of the CNN and the rule-based approach on datasets not used for training. For cones of the same type as the ones presented during training, both approaches provide similar results, giving an accuracy of around 96% for the ones close-by. However, the difference between the two arises when different cone types are used during testing. In the competition in Germany, the cones have an *FSG* sticker that results in different point cloud intensities. The rule-based approach shows a larger number of misclassifications, whereas the CNN is hardly affected by it. This supports our initial hypothesis that the CNN is more reliable and generalizes better than its rule-based counterpart.

[2]www.amzracing.ch

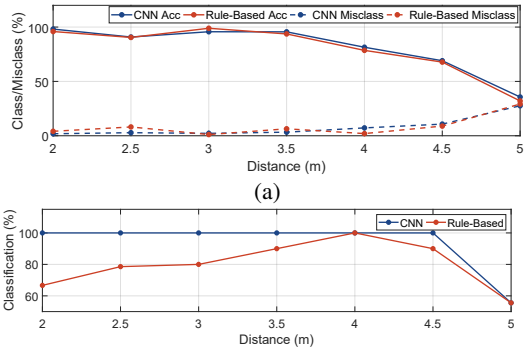


Fig. 8: Classification performance of the CNN and the rule-based approach when (a) using the same cones as in training and (b) different ones.

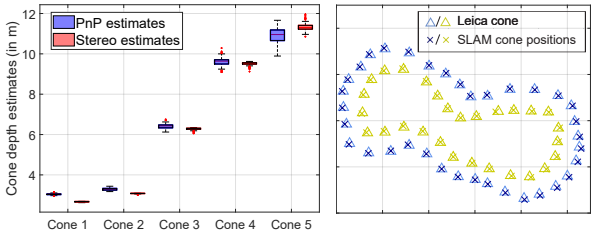


Fig. 9: (a) Box plots of depth estimates of five cones obtained from the left camera's image via PnP and via the stereo approach using triangulation (b) Resulting map computed in real-time using fastSLAM 2.0 implementation in the mapping phase, compared to a ground truth measurement using the Leica Totalstation. The grid size is 10m x 10m.

These figures also show the significant drop in the accuracy when the distance is around $5\,\mathrm{m}$, justifying the decision to cap color estimation at this distance. Most of the cones at such distances are labelled *unknown* which results in reduced classification accuracy, but not increased misclassification because a blue cone is not labelled yellow and vice-versa. This reduces the number of false color estimates, thus ensuring the robustness of the system.

B. Visual Cone Detection and Stereo Pose Estimation

Figure 9a compares the depth estimates of cones obtained from the PnP algorithm to those obtained through triangulation. Multiple measurements of the same scene are taken to illustrate the variance of the estimates. It can be observed that the variance is reduced significantly by using triangulation, especially for cones that are far away. However, due to the fact that disparity decreases with distance, the position estimates' accuracy drops at larger distances. Hence, the maximum depth estimate provided by the stereo setup is limited to $10\,\mathrm{m}$.

Figure 9b shows a map generated by SLAM for a $100\,\mathrm{m}$ long track using only the estimates from the stereo camera.

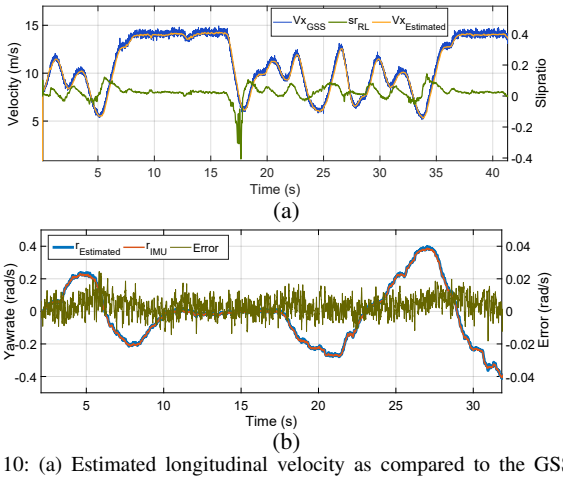


Fig. 10: (a) Estimated longitudinal velocity as compared to the GSS and (b) yaw rate estimates compared to the IMU.

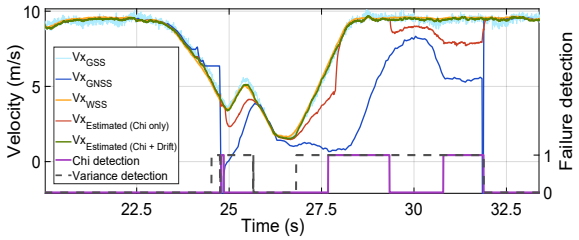


Fig. 11: Failure of the GNSS (cyan) is detected by the drift detection (dashed grey). $Vx_{Estimated(chi\ only)}$ (red) is the estimated velocity using the chi test (violet) which shows that the chi test without drift detection is unable to detect the complete sensor failure.

A RMS landmark error of $0.25\,\mathrm{m}$ is achieved, which is close to the accuracy achieved when using LiDAR estimates and enough to finish the race in case of a LiDAR failure.

C. Velocity Estimation

The redundancy of the velocity estimator is analyzed by simulating sensor failures and comparing velocities to ground truth (GT) information based on the GSS data. Figure 10a shows the estimated velocity without the GSS compared to GT. The RMSE is $0.14\,\mathrm{m\,s^{-1}}$. It can be seen that the velocity estimate is accurate even when the wheels slip a lot. The distance between the position of the car obtained by integrating the estimated velocity and the GPS position is less than $1.5\,\mathrm{m}$ over a $310\,\mathrm{m}$ long track, which results in a drift of less than 0.5 %.

Figure 10b compares the estimated yaw rate without IMU to that of the IMU, wherein it is evident that the predicted yaw rate almost converges to the true yaw rate. This implies that the car model is reliable and velocities can be accurately estimated even in the absence of the IMU.

Figure 11 shows the response of the filter to sensor failures. It can be observed that the chi-square based failure detection is able to reject the signal only when the failure is short-lived, whereas the drift failure detection is able to also discard continuous sensor failures. Using both techniques in conjunction ensures removal of most of the sensor failures.

D. Localization and Mapping

The accuracy of the SLAM algorithm is evaluated by comparing the generated map with the ground truth (GT) measurements from a Leica Totalstation. Figure 12 shows the mapping result for a $230\,\mathrm{m}$ long track using LiDAR data only. An RMSE of $0.2\,\mathrm{m}$ is obtained for both landmarks and the driven path. All cones' colors are correctly estimated. A performance comparison of fastSLAM 1.0 vs 2.0 is shown in Figure 13. The RMSE for a given amount of particles is significantly smaller for fastSLAM 2.0., implying higher map accuracy.

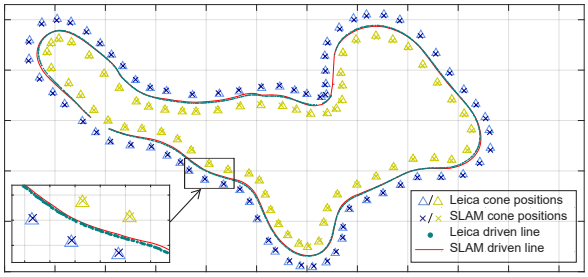


Fig. 12: Resulting map and trajectory computed in real-time during the mapping phase, compared to ground truth measurements from Leica Totalstation. The grid size is $10\,\mathrm{m} \times 10\,\mathrm{m}$ for the whole map and $2\,\mathrm{m} \times 2\,\mathrm{m}$ for the zoomed in image.

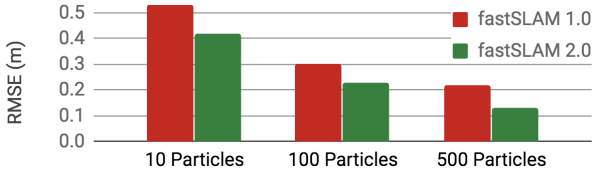


Fig. 13: RMS landmark error comparison of fastSLAM 1.0 vs 2.0 for different number of particles against ground truth from a Leica Totalstation.

V. Conclusion

This paper presented the approaches developed to ensure reliable operation of an autonomous race car by introducing redundancy into the perception and state estimation pipelines. It has been shown that accurate color estimates can be obtained from LiDAR using the intensity signature of the point clouds, and accurate positions can be estimated from cameras using prior knowledge of objects. Additionally, we have demonstrated accurate velocity estimation during high wheel slip and under single-sensor failure, captured by the failure detection module. As future work, it would be interesting to investigate whether the performance of the failure detection module can be improved by performing post-state analysis. Finally, the fastSLAM 2.0 algorithm has been adapted to map and localize in real-time using the output of either one or both perception systems. Extensive testing shows that the algorithms generalize well to unseen environments, even under sensor failure, thus paving the way for autonomous racing close to the limits of handling.

Acknowledgment

The authors thank the AMZ Driverless team for their sustained hard-work and passion, as well as the sponsors for their financial and technical support. We also express our gratitude to Marc Pollefeys, Andrea Cohen, and Ian Cherabier (CVG Group, ETH Zürich) for their support throughout the project.

References

[1] Sanjiv Singh. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Nov. 2009.

[2] J. Funke et al. "Up to the limits: Autonomous Audi TTS". In: *2012 IEEE Intelligent Vehicles Symposium*. June 2012, pp. 541–547.

[3] A. Liniger, A. Domahidi, and M. Morari. "Optimization-Based Autonomous Racing of 1:43 Scale RC Cars". In: *ArXiv e-prints* (Nov. 2017). arXiv: 1711.07300.

[4] Mark Harris. *The 2,578 Problems With Self-Driving Cars*.

[5] H. Cho et al. "A multi-sensor fusion system for moving object detection and tracking in urban driving environments". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. May 2014, pp. 1836–1843.

[6] C. Premebida et al. "Pedestrian detection combining RGB and dense LIDAR data". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2014, pp. 4112–4117.

[7] Q. Li et al. "A Sensor-Fusion Drivable-Region and Lane-Detection System for Autonomous Vehicle Navigation in Challenging Road Scenarios". In: *IEEE Transactions on Vehicular Technology* 63.2 (Feb. 2014), pp. 540–555.

[8] Miguel de la Iglesia Valls et al. "Design of an Autonomous Racecar: Perception, State Estimation and System Integration". In: *CoRR* abs/1804.03252 (2018).

[9] N. Bezzo et al. "Attack resilient state estimation for autonomous robotic systems". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2014, pp. 3692–3698.

[10] Steven Scheding et al. "An Experiment in Autonomous Navigation of an Underground Mining Vehicle". In: 15 (1999), pp. 85–95.

[11] K. C. H. Yang, J. Yuh, and S. K. Choi. "Fault-tolerant system design of an autonomous underwater vehicle ODIN: An experimental study". In: *International Journal of Systems Science* 30.9 (1999), pp. 1011–1019.

[12] Formula Student Germany. *FS Rules 2018 v1.1*.

[13] M. Himmelsbach, F. v. Hundelshausen, and H. -. Wuensche. "Fast segmentation of 3D point clouds for ground vehicles". In: *2010 IEEE Intelligent Vehicles Symposium*. June 2010, pp. 560–565.

[14] Bertrand Douillard et al. "On the segmentation of 3D LIDAR point clouds". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2798–2805.

[15] Renaud Dubé et al. "Segmatch: Segment based place recognition in 3d point clouds". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 5266–5272.

[16] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 6517–6525.

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004.

[18] Luc Van Gool Ankit Dhall Dengxin Dai. *Real-time 3D Traffic Cone Detection for Autonomous Driving*. Report. ETH Zürich, 2019.

[19] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60 (2004), pp. 91–110.

[20] Hans B Pacejka. *Tire and Vehicle Dynamics*. Butterworth-Heinemann, 2012.

[21] Advanced Navigation. *Section 10.4 Spatial Dual Reference Manual*. 2017.

[22] Kistler Group. *Section 7.8 Correvit non-contact optical sensors instruction Manual*. 2017.

[23] Zongwen Xue and Howard Schwartz. "A Comparison of Several Nonlinear Filters for Mobile Robot Pose Estimation". In: *2013 IEEE International Conference on Mechatronics and Automation*. 2013.

[24] Mehmet Ugras Cuma and Tahsin Koroglu. "A comprehensive review on estimation strategies used in hybrid and battery electric vehicles". In: *Renewable and Sustainable Energy Reviews* (2014).

[25] Moustapha Doumiati et al. *Vehicle Dynamics Estimation using Kalman Filtering*. ISTE Ltd and John Wiley & Sons, Inc, 2013.

[26] Michael Montemerlo et al. "FastSLAM 2.0 : An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges". In: *Proceedings of the 18th international joint conference on Artificial intelligence*. 2003, pp. 1151–1156.

[27] Michael Montemerlo and Sebastian Thrun. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*. 1st. Springer Publishing Company, Incorporated, 2010.

[28] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[29] J. Nieto et al. "Real time data association for FastSLAM". In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 1. Sept. 2003, 412–418 vol.1.

[30] Thomas P. Minka. *Bayesian Inference, Entropy, and the Multinomial Distribution*. 2003.