.02982

DMB

DATA MANAGEMENT
AND
BIOMETRICS

STUDY OF SELF-DRIVING FUNCTIONALITY: FROM SDC TO LANE BOUNDARY DETECTION-BASED IMPLEMENTATION IN SMALL-SCALE KARTS

Raj Kumar Ashokan

MASTER'S ASSIGNMENT

October, 2024

2024DMB0008
Data Management and Biometrics
EEMathCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

ROBOTICS AND MECHATRONICS

STUDY OF SELF-DRIVING FUNCTIONALITY: FROM SDC TO LANE BOUNDARY DETECTION-BASED IMPLEMENTATION IN SMALL-SCALE KARTS

R.K. (Raj Kumar) Ashokan

MSC ASSIGNMENT

October, 2024

072RaM2024
Robotics and Mechatronics
Computer Science/Data Management & Biometrics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE
UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Self-Driving Challenge: Implementation of Vision-Only Based Autonomous Driving in Karts

Raj Kumar Ashokan   Gayathri Dhanapal

***Abstract*—This study is based on our participation in the Self Driving Challenge 2023 edition, which was aimed at the study of basic autonomous functionality behaviour in cars. The purely vision-based, classical autonomous driving approaches implemented during the challenge are discussed here. This includes an unsuccessfully attempted monocular visual odometry based approach, in which relative localization was successfully obtained using feature extraction, feature matching, and 2D-2D motion estimation but absolute world scale could not be successfully retrieved. A lane boundary marker detection based approach was then successfully implemented and utilized at the challenge. This approach enabled the provided electric go-kart to autonomously traverse a distance of approximately 1 km. The algorithm processing speed was 30 FPS in real-time. This approach fetched us the runner-up position at the challenge. The observed outcome at the challenge is presented including noted undesirable behaviours. However, the behaviours could not be analyzed or studied owing to the short development stint of the challenge. The shortcomings at the challenge are also identified for both the approaches along with the need for follow-up study. A video of our demonstration of the autonomous driving at the SDC finale event can be found at: https://tinyurl.com/4ycet5de.**

***Index Terms*—self driving challenge, RDW, autonomous cars, monocular visual odometry, relative localization, motion estimation, lane boundary detection, steer estimation**

I. Introduction

Self-driving car technology has been making great strides towards becoming a reality and has been changing day to day lives in terms of road safety [1], ease in mobility, increased travel comfort, reduced emission and pollution levels [2], improved transport inter-connectivity etc. since its advent. Vehicles are increasingly equipped with advanced sensors, vision and control systems, providing them with autonomous capabilities [3] [4]. It has become inevitable to further probe into this field and research into the latest advancements in order to expand the knowledge in smart mobility.

In line with this, being the regulatory organization in approving cars for use on public roads, the Netherlands Vehicle Authority (RDW) has been organizing an annual competition, 'Self Driving Challenge (SDC)', since 2019. The RDW organizes this challenge with the futuristic goal of preparing itself for expanding its knowledge about autonomous vehicles, especially cars, and about the complex choices those vehicles make [5]. SDC being an open challenge for the student teams in The Netherlands, we participated in the SDC 2023 edition as a part of the team from the University of Twente. The main aim of this edition was to build a software stack to autonomously drive a lap as fast as possible at the specified track using an electric go-kart that is provided by the RDW.

Fig. 1: University of Twente team at SDC2023. Image credit: RDW/Self Driving Challenge

Autonomous racing on karts provides a valuable testing field for algorithmic approaches related to autonomous driving. As this field is emerging and relatively new, a direct transfer of autonomous racing software to the autonomous passenger cars has not yet been accomplished [6]. However, increasingly, more autonomous racing challenges are organized using karts, such as the EV Grand Prix Autonomous Challenge [7] and Formula Student Driverless competitions [8]; these challenges induce valuable research that can be scaled up to passenger cars. Therefore, study of basic autonomous functionality behaviour using karts can be a good starting point in the study of the same in cars, aligning with the motive of the SDC [5].

Each participating team at the SDC had limited access to two identical electric go-karts, equipped with the required hardware, and thus the extent of the challenge differed only in the development of the autonomous software and interfacing with the hardware [9]. The whole challenge took place at the TT-Junior track, at Assen in the Netherlands, which is a racing circuit of 1 km length and varying width. Being a single-lane circuit with boundary markers on both sides, it also contained turns, intersections, splits, crossings, curbs and inner loops.

A total of six teams participated in the edition and our team secured the second prize. At the end of the challenge, our developed code was able to provide autonomous functionality to the provided go-kart, but extensive testing could not be performed to evaluate its autonomous behaviour. This extensive testing and evaluation was done as a follow-up study at the University of Twente, by using a rebuilt small-scale kart. This follow-up study is not discussed in this paper. The approaches we used for participating in the challenge and the performance at the challenge is primarily discussed here. Overall, the objective of this Master's thesis work was to implement and study the behaviour of basic autonomous functionality in cars using karts and small-scale karts. Fig. 1 shows the SDC electric go-kart during one of its autonomous manoeuvres demonstrated by the University of Twente team.

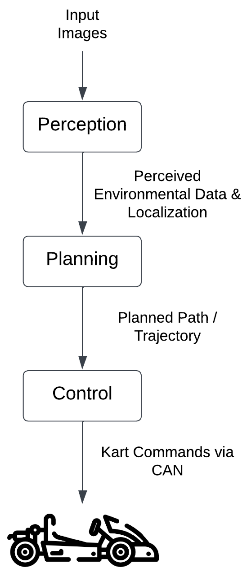

Input Images
Perception
Perceived Environmental Data & Localization
Planning
Planned Path / Trajectory
Control
Kart Commands via CAN

Fig. 2: Basic autonomous driving pipeline

The pipeline for a vehicle to drive itself autonomously from a point A to point B includes three major modules - perception, planning and control as in Fig. 2. Perception is the ability of a vehicle to perceive its surroundings and to know its own position in the environment using the data from its sensors. Planning is the process of generating the best path for the vehicle to traverse and reach its destination, based on the perceived environment taking into account the dynamic capabilities of the vehicle, presence of obstacles etc. Control is the process of converting the intended decisions into actions by sending commands to the actuators to obtain the desired movement [10].

In the SDC kart, cameras were meant to be the primary sensor setup, limiting the scope to visual perception based systems. Obstacle avoidance was not included in the scope and hence is not considered in the entirety of this work. One of the challenging perception tasks for an autonomous vehicle is estimating its current ego-pose or localization [11] [12]. Visual-based localization systems can be broadly based on traditional, learning or hybrid approaches. The state-of-the-art learning-based or hybrid approaches require a lot of data and processing power [6] [13] for considerable performance. However, during the SDC, owing to the limited processing capabilities of the kart, we could not rely on that as the primary method. Nevertheless, two other members of our team were trying to develop an end-to-end steering prediction, which was not progressively fruitful during the competition and, therefore, is not discussed here.

Considering traditional approaches, Visual Odometry (VO), which is a process of estimating the translational and rotational movements of the camera using images, is often used for localization in autonomous vehicles [14] [15]. Therefore, we initially considered monocular visual odometry for localization, as the kart did not include a stereo setup. The involved steps are: feature extraction, feature matching, outlier rejection, and relative pose estimation. Relative ego localization was successfully obtained using this approach; to integrate the absolute scale, an image retrieval-based method was tried to be implemented, but in vain. This involved constructing a database of compressed geo-tagged images and fetching matched images using image retrieval. Thus, the scale factor issue of the monocular camera could not be successfully resolved utilizing any other additional information, resulting in unsuccessful localization. For the planning and control steps in the pipeline, a waypoint follower utilizing a geometric lateral controller was planned to be implemented in case of successful localization using odometry.

Therefore, in the later part of the challenge, we a adopted lane boundary detection-based road following approach, which is another commonly used approach; this was the approach we used for the final race. The steps involved in the perception module here includes feature extraction, detection of lane boundary markers, and lateral localization relative to the lane markings. In the planning module, steer angle is calculated in order to generate the trajectory for lane following, i.e., for the kart to be positioned at the center of the lane. The lateral control module executes the movements, including error handling process, to maintain the planned path.

Traditionally, the study of related works would be carried out at the beginning of the research. However, as this work directly started with the development phase due to the stringent schedule of the competition, the study of related works was done at the beginning of the follow-up study and is not elaborated in this paper. The contributions discussed in this paper focuses on the design-development approach and the implementations carried out to satisfy the requirements of the challenge, which were primarily given by:

- The kart should begin from the start position and run autonomously through the track. The teams would not compete against each other at the same time, but one after the other.
- Each team would get a time-slot of 15 minutes for multiple trials. A trial would be immediately considered disqualified if the kart either goes off-track or even touches either of the lane boundary markers with one of its tyres.
- After disqualification, a new trial would begin again from the start position, only within the provided time-slot.
- The kart should follow the right path at the intersections or crossings.

The team that makes the most metres on the track in the shortest lap time possible, satisfying the above requirements

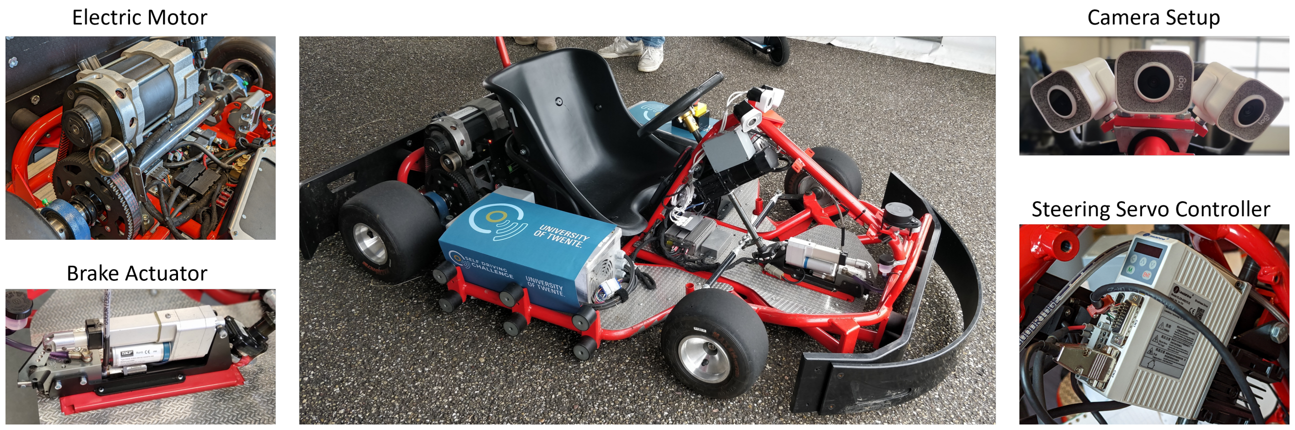


Fig. 3: The SDC electric Go-Kart

would emerge as the winner. More details of the challenge can be found at [9].

The paper is further structured as follows: Section II describes the hardware used. Section III talks about the data collection process. Section IV details the approaches used. Section V discusses the results and the performance at the challenge. Section VI concludes the work with highlights and shortcomings at the challenge, and talks about the follow-up study.

II. Go-Kart

The parts of the electric go-kart relevant to the challenge, as shown in Fig. 3, include the computing unit (a 16GB Intel i5 processor with no GPU), actuators, and sensors, apart from the chassis. The actuation of the go-kart is primarily made via throttle, steering, and braking. The original version of the go-kart consisted of interface modules such as steering wheel and pedals which were human driven. For the sake of autonomous driving, a servo motor (for steering) and a linear actuator(replacing the braking module), which are controlled via ECUs, were integrated in the provided kart. Communication between the ECUs is carried out over a CAN (Controller Area Network) bus, emulating the standard communication system within a conventional car [16]. The kart comes equipped with a 3.5 kW electric motor for longitudinal actuation with maximum speed modes of either 5, 15, 30 or 60km/h. Apart from autonomous control capability, manual control via a wired Xbox controller was also possible.

Existing research works concerning autonomous driving involve fusion of data from sophisticated sensors such as the 3D LiDAR, RADAR, GNSS, IMU, encoders, and cameras in their perception module [17] [18]. In contrast to this, the SDC kart was equipped with a basic sensor suite with three regular USB web-cameras and a 2D planar LiDAR. The cameras were clamped together at a height of 60 cm from the ground in the front part of the kart and had negligible overlap between their field of views. The 2D planar LiDAR, also placed at the front part of the kart, primarily finds its usage in obstacle avoidance task and hence was of little use to us. For safety reasons, an emergency transmitter-receiver pair was interfaced for easy manual intervention.

III. Dataset

Data collection from all the three cameras was performed in parallel, by manually driving the kart throughout the track in lane-centered, lane-edged, and zig-zag manoeuvres. Data was captured in three different speed modes, during different times of the day, and by different drivers. Cameras were calibrated to obtain the intrinsics. In addition to the image recordings, the set of throttle, steer, and brake commands given to the actuators via the CAN network were also recorded in the form of a CSV file. Both the CSV and image data are time synchronized utilizing the concept of multi-threading. This is significant as it facilitates better understanding and correspondence between images and actuator commands, especially during algorithm development and analyses.

IV. Approaches

A. *Visual Odometry Based Approach*

1) Feature Extraction: The overview of the processes in the initial perception module for monocular VO based approach is as shown in Fig. 4. In this module the center camera images are utilized to obtain the kart's localization. Extraction of features or keypoints in the image is the primary significant step in the process. Features can be extracted using various methods like SURF, Harris corner detector, ORB, FAST etc. ORB (Oriented FAST and Rotated BRIEF) [19] can detect and describe (vectors of size 32) more features, that are invariant to scale, rotation, and small affine changes, quickly than many such feature detector-descriptors [12] [20]. Hence ORB was initially chosen for this step primarily for the sake of efficient computation. An example image with features extracted using ORB is as shown in Fig. 5a. Later in the further stages of the pipeline, when the results of the obtained motion estimation were not as expected (discussed in the Results section), another detector-descriptor known as SIFT (Scale Invariant Feature Transform) [21] was finally used. SIFT is a more accurate method, which detects stable features that are robustly invariant to scale, rotations or small affine changes than ORB [17], and provides descriptors (vectors of size 128). But this comes with a higher

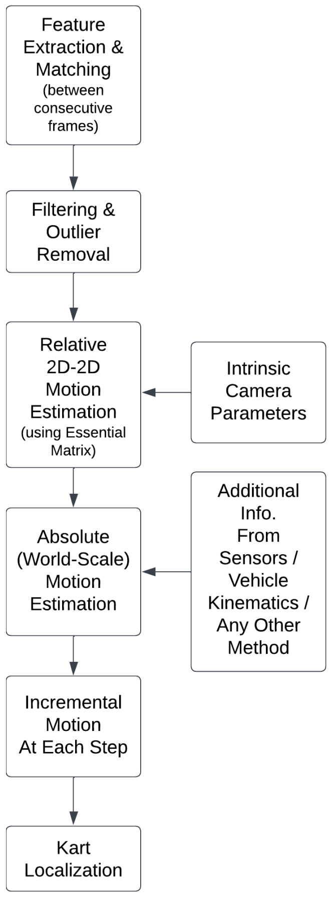
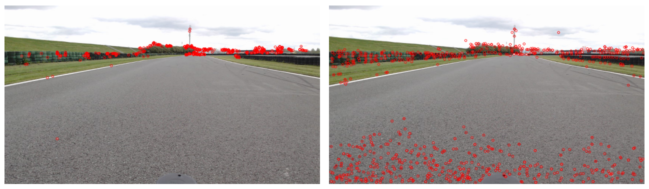

Feature Extraction & Matching (between consecutive frames)
Filtering & Outlier Removal
Relative 2D-2D Motion Estimation (using Essential Matrix)
Intrinsic Camera Parameters
Absolute (World-Scale) Motion Estimation
Additional Info. From Sensors / Vehicle Kinematics / Any Other Method
Incremental Motion At Each Step
Kart Localization

Fig. 4: Processes in the perception module

computational cost [20]. Features extracted using SIFT for the same example image are as shown in Fig. 5b.



(a) Using ORB **(b)** Using SIFT

Fig. 5: Sample outputs showing extracted features

2) *Feature Matching:* Secondly, feature matching between the extracted SIFT features of two consecutive images was performed using a Brute Force(BF) matcher. It considers a descriptor in one image and tries to find a match among all the descriptors in the other, based on the smallest distance. For binary string-based descriptors like ORB, the Hamming distance is considered, whereas for SIFT, the L2 Norm works good [22].

For each feature, two best matches were retrieved and a distance ratio thresholding based on [21] was performed, to filter out unreliable matches. This requires the closest match to be significantly better than the second closest match by checking the ratio of their distances. This ratio threshold was varied from 0.6 to 0.9 to see which threshold was more reliable. Feature matching using BF in two consecutive frames is visualized as shown in Fig. 6. BF-based matcher is highly accurate but requires high computation time. FLANN-based matcher(Fast Library for Approximate Nearest Neighbors) is another technique that finds matches by approximating the nearest neighbours instead of computing them exactly. This is optimized and quicker than BF for larger datasets [22] [23]. Therefore, for the sake of efficiency, this matching technique was also attempted but the quantity of matches obtained were significantly reduced and thus FLANN was not chosen over BF.

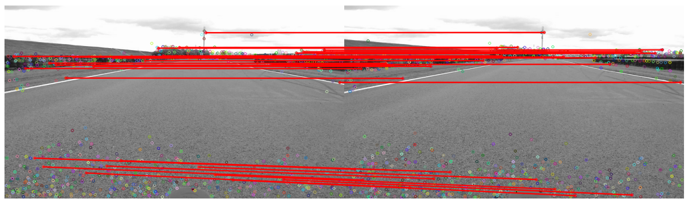


Fig. 6: Matched features between sample consecutive frames. Only a few matches are shown for illustration

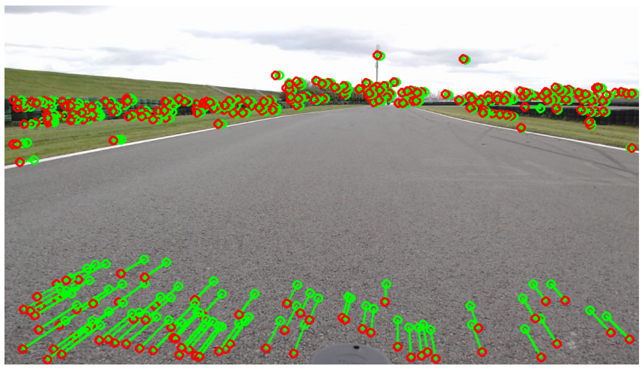


Fig. 7: Estimated motion between sample consecutive frames

Another commonly used technique to improve efficiency is to perform feature tracking instead of matching. Feature tracking using Lucas-Kanade optical flow method was tried. Features generated in a frame are searched for in the consecutive frame using search windows and a pyramid level search approach. Features that are not successfully tracked are dropped; new features are regenerated if the number of retained features drops below a set limit. Again owing to improper results (as discussed later), feature matching using BF was the final chosen approach to find correspondences between consecutive frames.

3) *Relative Motion Estimation:* From the obtained correspondences or matches between the two consecutive frames and the intrinsic parameters of the camera, 2D-to-2D camera motion was estimated by computing the Essential matrix. This step also incorporates additional outlier removals using RANdom SAmple Consensus (RANSAC), as the filtered matches might still contain outliers. RANSAC iteratively selects random subsets of data and fits a model hypothesis, identifying inliers that align with the model. Assuming known intrinsics, the essential matrix encodes the epipolar

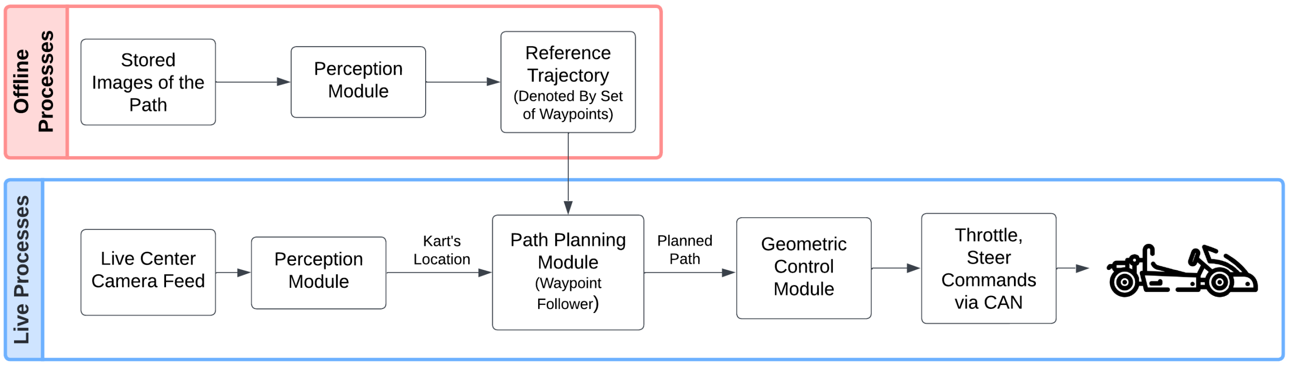

Offline Processes
Stored Images of the Path
Perception Module
Reference Trajectory (Denoted By Set of Waypoints)
Live Processes
Live Center Camera Feed
Perception Module
Kart's Location
Path Planning Module (Waypoint Follower)
Planned Path
Geometric Control Module
Throttle, Steer Commands via CAN

Fig. 8: Overview of the offline and the live processes for the monocular VO-based approach

geometry between two camera frames. This is further decomposed into the relative rotation (R) and translation (T) using Singular Vector Decomposition(SVD) and chirality check, resulting in relative pose between the consecutive frames [24].

The estimated motion between features of two consecutive frames is as depicted in Fig. 7. By accumulating this incremental motion at each step, the trajectory followed by the camera can be generated. Notably, this trajectory is not in absolute scale i.e., real-world units. As the camera is rigidly attached to the kart, the motion of the camera corresponds to the motion of the kart. Resolving this scale issue using any other additional information, trajectory in absolute scale can be generated. By using the collected dataset, a reference trajectory can be generated in this manner;during the live run, a waypoint follower can be used to follow this trajectory by estimating and accumulating the camera poses over time. The overview of the offline and the live processes for the monocular VO based approach with respect to the perception-planning-control pipeline is as shown in Fig. 8.

4) Absolute world-scale estimation: Subsequently, for obtaining the absolute scale, we tried to apply a global localization method partly as in [11], wherein the authors try to obtain the global pose using image retrieval method and a mapping database of geo-tagged images.The purpose of image retrieval is to fetch images similar to the query image from the database. In order to reduce the computational cost of this retrieval, it is necessary to have compact image representations in the database. For creating this offline database, the first essential step is to build a visual vocabulary from the images. A complete set of approximately 14000 reference dataset images was used for this purpose and their SIFT feature descriptors were extracted. Using these descriptors as input data, a k-means classifier was trained to partition the descriptors into 64 clusters. The centres of these 64 clusters are representative feature descriptors and hence form the visual vocabulary.

Secondly, a residual error was calculated between every descriptor of the image assigned to the k-th cluster and the center of that cluster. These residual errors were summed up, giving a total residual error for each cluster. Computing this

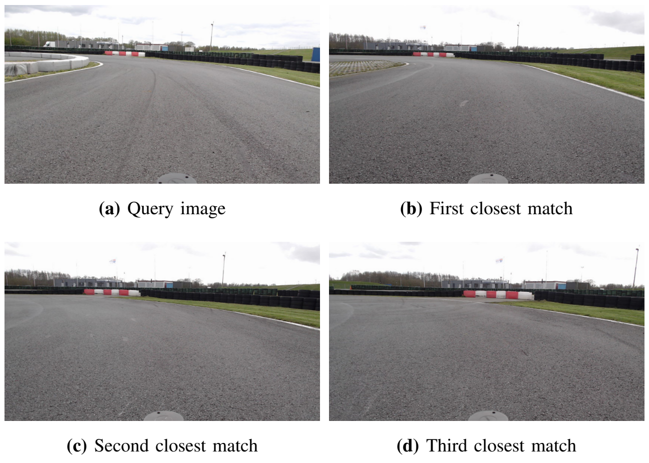

(a) Query image (b) First closest match (c) Second closest match (d) Third closest match

Fig. 9: Sample outputs of global localization module

for all the 64 clusters and stacking up, a 64×128 matrix (as the SIFT descriptor size is 128-dim) was obtained for each image. This matrix was L2 normalized to obtain a Vector of Locally Aggregated Descriptors(VLAD) matrix, which is detailed in [25] and [26]. VLAD matrices of all the images in the considered dataset was generated to form a mapped database. During the live run, in order to fetch the best match for a query image, a BallTree nearest neighbour algorithm trained on this mapped database is used.

The image fetched with the lowest distance is chosen as the best match. A sample query image of the Assen track, along with its three closest matches retrieved via this method is as shown in Fig. 9. Visibly, the retrieved matches are properly indicative of the location despite the fact that majority part of the image contained only the road surface, sky, and other less distinctive information.

For tagging the location corresponding to the mapped database images, in [11], a GPS is used in the offline process and a 3D LiDAR is also used to obtain the 3D coordinates of the features. Using this information and a particle filter, the relative world position of the query image, with respect to the retrieved three best matches can be obtained. In our case, we tried to use a constant velocity assumption or to

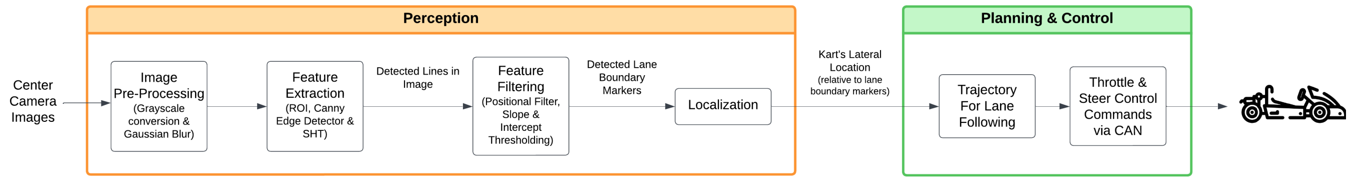
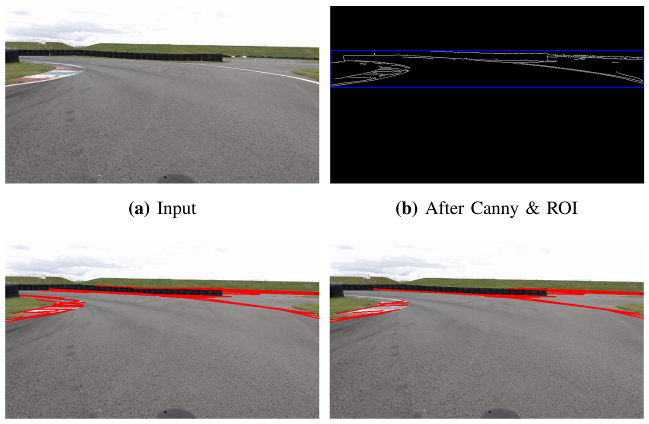

Perception
Planning & Control
Center Camera Images
Image Pre-Processing (Grayscale conversion & Gaussian Blur)
Feature Extraction (ROI, Canny Edge Detector & SHT)
Detected Lines in Image
Feature Filtering (Positional Filter, Slope & Intercept Thresholding)
Detected Lane Boundary Markers
Localization
Kart's Lateral Location (relative to lane boundary markers)
Trajectory For Lane Following
Throttle & Steer Control Commands via CAN

Fig. 10: Steps involved in perception, planning and control modules of lane boundary detection based road following approach



(a) Input **(b)** After Canny & ROI

(c) After SHT **(d)** After Positional Filter

Fig. 11: Sample outputs at intermediate steps

tag the location using any other independent positioning device. However, unfortunately, we could not succeed with this step, owing to cancellation of some of the track days and unforeseen practical hindrances during the test slots.

Furthermore, as only a couple of track days were pending then, which would not be sufficient to do the testing and to implement the following adaptations required for this approach, this approach was dropped; an alternative lane detection-based approach, was attempted to be implemented in the remaining days to the challenge, which is detailed as follows.

B. Lane Boundary Detection Based Approach

This approach builds on the detection of the lane boundaries, using the center camera images as the primary feed. Based on this, the steering angle for lateral control is calculated.

1) Pre-Processing and Feature Extraction: Initially, as the first step in perception, we tried to project the input images using Inverse Perspective Mapping to correct for the perspective distortion [27]. However, in our case, the transformed image did not contain meaningful lane information as the road was very wide. The other option was to perceive using the camera perspective view.

For line detection, grayscale conversion and Gaussian blurring were typically carried out on the input feed to reduce the computation costs and the noise resulting from low camera height [2] [28]. Following this, lane boundary features were detected using Canny edge detector, as it preserves necessary structural information while removing unwanted intense data. In the resulting image, geometric model fitting Standard Hough Transform(SHT) was applied on a chosen Region of Interest(ROI) to characterize the line segments belonging to the road boundaries. The obtained detected lines often included irrelevant lines or false positives as in Fig. 11c; for removing these, a series of line filtering and refinement steps were applied next.

2) Feature Filtering and Localization: Consequently, lines that could be a potential left boundary but present in the right half of the image and vice versa were removed, using a positional filtering that checks the sign of the slope and the image co-ordinates, as in Fig. 11d. This was based on the analysis that unless the kart is off-track, at least one-third portion of the left boundary falls in the left half of the image and similarly for the right boundary. Filtering based on the geometric properties like slope thresholding and intercept thresholding was performed next. Different sample images showing the impact of the above steps are shown in Fig. 12a - Fig. 12d.

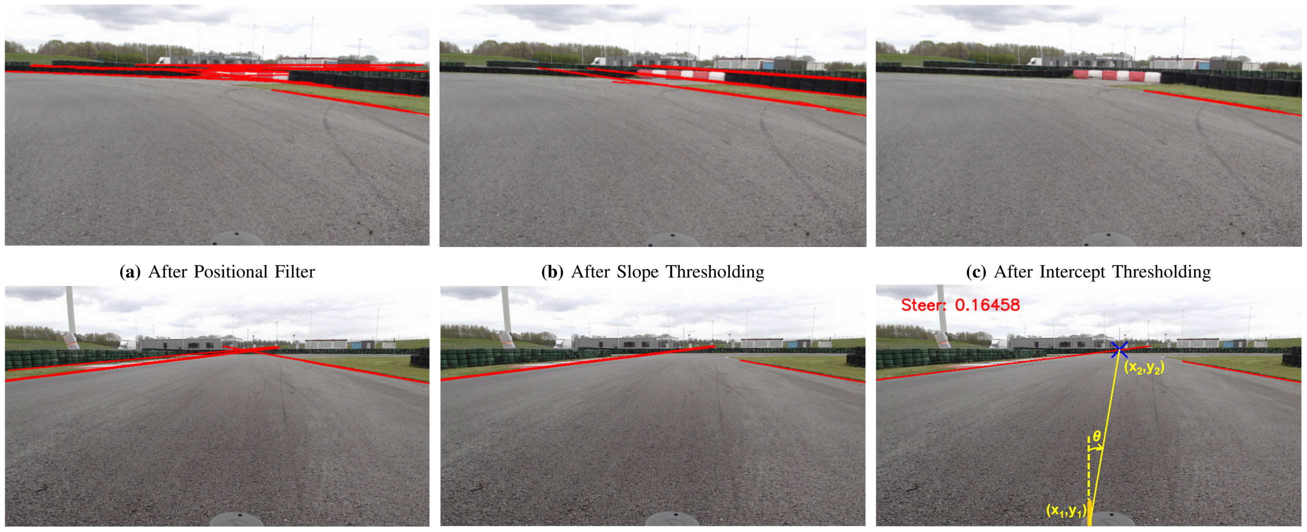
Subsequently, the obtained lines were classified into potential left and right boundaries using the sign of the slope. Owing to the camera's perspective and the wide roads, to avoid false detections of the close-by road edges and baseline of tyre barrier walls, a clustering operation was then carried out by selecting only the lines with a intercept value greater than the mean intercept value on each side as in Fig. 12e. Merging the resulting individual line segments together, their end coordinates were averaged on each side to form the final left and right lane boundaries. If no lines were present during filtering process, next frame was grabbed and processed.

3) Lane Following: Next, in the planning module, steer value was computed for the kart to run in the lane's center. This was calculated using the point of intersection of the detected lane boundaries [13] [28]; this point of intersection denotes the desired heading as shown in Fig. 12f, and is used in the steer(ϕ) computation as given by:

$$\theta = \tan^{-1}\left(\frac{y_1 - y_2}{x_1 - x_2}\right) - 90^\circ \tag{1}$$

$$\phi = \frac{\theta}{S_{max}} \begin{cases} > 1 \rightarrow \text{right steer} \\ < 1 \rightarrow \text{left steer} \end{cases} \tag{2}$$

where (x_1, y_1) and (x_2, y_2) are the coordinates of the image base centre point and the point of intersection of the detected lane boundaries respectively; (θ) denotes the deviation of the vehicle's heading, S_{max} denotes the maximum possible steer of the kart.

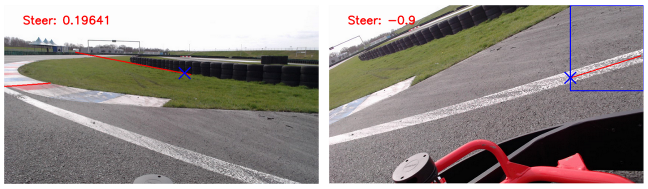


(a) After Positional Filter **(b)** After Slope Thresholding **(c)** After Intercept Thresholding

(d) After Intercept Thresholding **(e)** After Clustering **(f)** After Merge & Steer Calculation

Fig. 12: Sample outputs at intermediate steps (cont.) for two scenarios (First scenario - Row 1; Second scenario - Row 2)

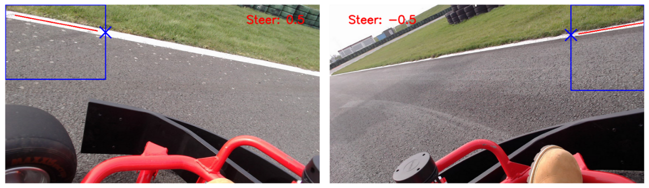
Crucially, in cases where either the left or right boundary was not present or not detected, we considered the extreme column of the image in that particular side as the detected boundary. Mostly such cases were encountered in wide turns, where this manner of handling seemed to work. When both boundaries remain undetected, after a few frames of reusing the previous steer value, the kart comes to a halt. The steering values obtained thus and the fixed throttle values are provided to the actuators in the control module. With the intention to avoid abrupt turns of the kart, we used a slower throttle value during the turns.



(a) Perceived from center camera **(b)** Perceived from right camera

Fig. 13: Need for off-track avoidance

4) Off-Track Avoidance: Nevertheless, due to unforeseen issues, the kart might run off-track. As seen in Fig. 13, despite all the filtering, presence of curbs generates a critical undesired steer at the boundary. Hence we introduce a significant fail-safe 'off-track avoidance' module using the left and right feed. For this, the zig-zag and lane-edged run images were analyzed to identify an optimal Region of Search(RoS), each in the left and right feed, in which boundary lines will appear only when the kart nears it. During live runs, this RoS will be checked for presence of lines and a steer calculation is proportionally derived based on the closeness to the boundary. Line detection was performed using the same steps as the center camera. As these two cameras were downward facing and close to the boundary, strong noiseless edges were detected thereby eliminating the need for major filtering operations.



(a) From left camera **(b)** From right camera

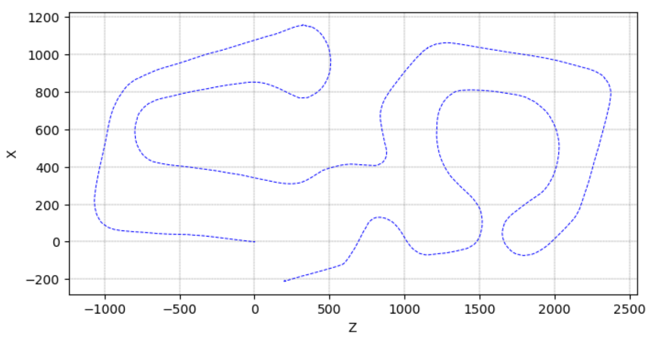
Fig. 14: Sample outputs from off-track avoidance module

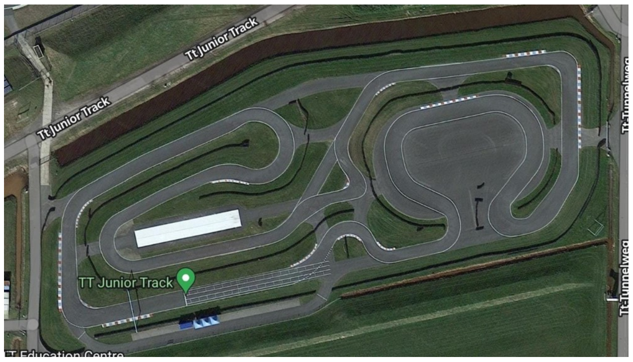
The point of intersection of this resulting final left or right lane boundary line with the inner vertical boundary of the RoS is found. As this point moves from the top to the bottom of the vertical boundary, in case of left camera feed, a positive steer value S_{Right} (ranging from 0 to 1) is proportionally devised to move the kart away from the left boundary; similarly a negative steer S_{Left} is devised for the right camera feed, demonstrated clearly as shown in Fig. 14. If no left or right boundary line is detected, then S_{Right} or S_{Left} value is zero respectively. This module simultaneously performs the check in both the cameras and outputs both S_{Left} and S_{Right}.

Combining this module along with the center camera module in a uni-modal sensor fusion, this module's outputs receive the highest priority. In case of non-zero values of either S_{Left} or S_{Right}, it is directly given to the kart to immediately steer it away from the boundary. If both S_{Left} and S_{Right} are zero, the steer from the center camera module is passed to the kart. If both are non-zero, we considered the S_{Right} to be given to the kart, specifically because of the rare false-positive scenarios that we faced due to the tire-markings on the tarmac on one side of the lane.

V. Results & Discussion

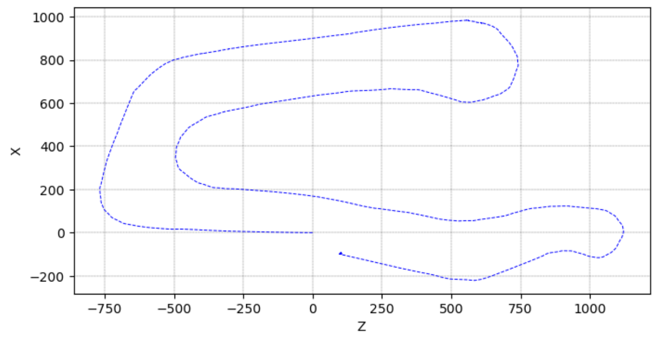
The results of the offline processes performed for both the VO and the lane boundary detection-based approaches are

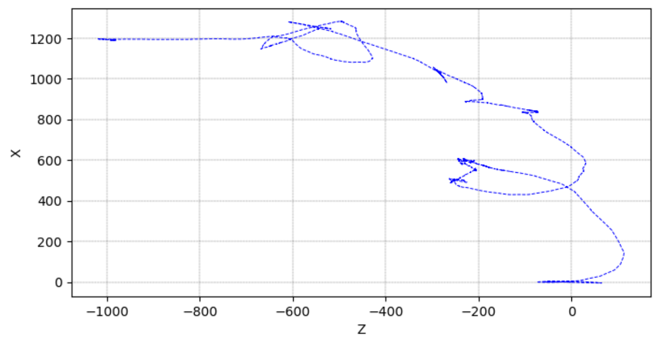

(a) SIFT+BF



(b) Original track from google maps

Fig. 15: Comparison of reference trajectory generated using SIFT+BF and the original track


(a) Using SIFT+BF


(b) Using ORB+BF

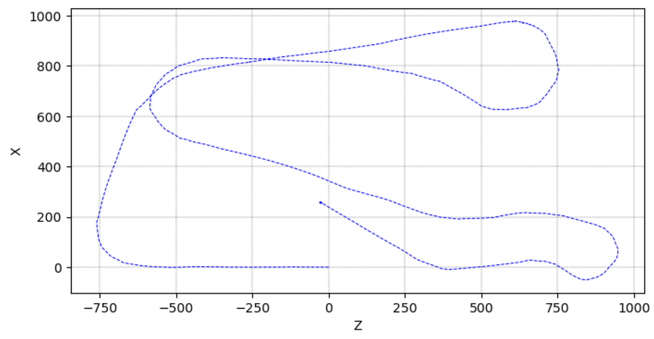

(c) Using SIFT+KLT

Fig. 16: Comparison of reference trajectories generated for first half of the racing track

discussed here in addition to the real-time performance of the kart at the challenge.

A. Visual Odometry Based Approach

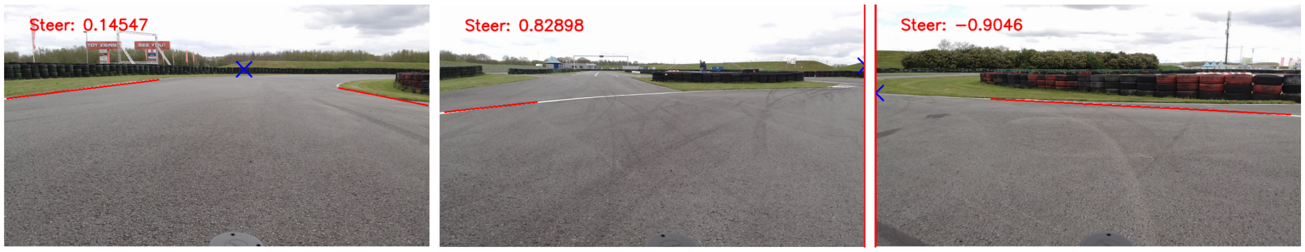
For this approach, the offline reference trajectory generated utilizing SIFT+BF, for a distance ratio threshold of 0.7, is as shown in Fig. 15a. Though this trajectory is scale ambiguous, the shape retrieved exhibits similarity with the original track as shown in Fig. 15b. However, it failed to achieve loop closure with slight drifts over time; this is consistent with the classic characteristic of VO process and is usually corrected using batch corrective techniques such as bundle adjustment [29]. For the sake of comparison, trajectories for half the track were generated using SIFT+BF, ORB+BF, SIFT+KLT (feature tracking) as shown in Fig. 16. ORB failed to perform, even for varying thresholds. This is presumably because SIFT features are generally detected across the entire image in a scattered manner, whereas, ORB features tend to concentrate more around corners, which is less applicable in our images as ground plane dominates the scene; this also resonates with the feature extraction step shown in Fig. 5a - Fig. 5b. In the shown sample image, the number of initial features extracted using ORB and SIFT were 499 and 903 respectively, and after distance ratio thresholding, the number of obtained features were 178 and 302 respectively.

In contrast to the ORB+BF method, SIFT+KLT method did not completely fail, but rather showed a drift accumulated at some parts of the trajectory. This is mainly due to build up of minor errors that occur while continuously re-estimating the position of the features. This is more significant when tracking is to be done for long sequences as in our case [30].

Thus, SIFT+BF performs better compared to using ORB+BF and SIFT+KLT. While tuning the optimal distance ratio threshold using SIFT, a lesser value of 0.5 or 0.6 filtered out too many features including inliers, resulting in a improper trajectory. On the contrary, a high value of 0.9 retained too many outliers in the matches, which largely remained even after applying RANSAC; this was again indicated by the improper shape of the trajectory obtained.

B. Lane Boundary Detection Based Approach

For lane boundary detection based approach, the final predicted steer results for some of the different scenarios are shown in Fig. 17. Owing to very short development span,



(a) Both lane boundaries visible **(b)** Only left lane boundary visible **(c)** Only right lane boundary visble

Fig. 17: Sample steer outputs for different scenarios

testing was initially carried out fully offline by assuming the possible problematic scenarios at different parts of the track. Possibility of simulation was ruled out due to rendering issues with the provided server and incorrect simulation map configuration. Real-time testing was performed directly at the pre-qualification and finale events. During the pre-qualification, in the initial runs, the kart drove autonomously but crossed weak-edged lane boundaries at two places. Tweaking the line detection parameters, the kart ran autonomously twice in the following runs, traversing the entire length in approximately 15 minutes at an execution speed of 30 FPS. However, it was exhibiting oscillating movements with left/right drags at some places of the track, specifically, at the turns. Notably, the kart also did not follow the path along the center of the lane in some segments of the track. Another significant point is that the off-track avoidance module seemed to work perfectly, with the kart staying on the track. On places where the kart neared the boundaries, a notable push was observed which steered the kart inwards, away from the boundary, right in time. Including such a fail-safe module is often overlooked by many, which was also noteworthy during most other teams' performance at the challenge.

At the finale slot, unfortunately, due to a malfunction, the initial version of the program (before the tweaking of the line detection parameters) had to be run, which made the kart to cross the weak-edged lane boundary at the same place as witnessed during the pre-qualification. As the slot duration was only 15 minutes, this was the only run that the kart made. Later the same day, when the malfunction was rectified and the correct program versions were used, the kart managed to run throughout the track twice autonomously in the first two speed modes. The kart exhibited the same behaviour as exhibited during the full autonomous runs performed at the pre-qualification. Since these runs were made outside the finale slot, our team won the second position.

VI. Conclusion

Overall, the VO based approach was unsuccessful because it could not obtain localization in absolute world scale, as no additional information could be employed; hence this approach could not be utilized for the challenge. On the other hand, the lane boundary detection based approach was implemented and succeeded at the challenge, despite the shorter development span. The kart managed to autonomously run a distance of approximately 1 km with a processing speed of 30 FPS. Still, no quantitative evaluation or analysis could be carried out, except for observing the kart's behaviour during the run, as no data was recorded for the sake of efficiency during the challenge. The reasoning behind the kart's observed oscillating behaviour could not be figured out. The reason for the the kart to follow a path which was deviated from the lane center could not be identified. Whether it was the center camera or the left-right pair that influenced the kart's behaviour could not be ascertained either, highlighting these as the shortcomings at the challenge. Despite the shortcomings, the approach shows potential to work. Similarly, if successfully implemented, VO based approach can be a more generalized one. Thus, the follow-up studies focus on overcoming the shortcomings arising from both the approaches used at the challenge and studying the autonomous behaviour in more detail.

References

[1] I. Kostavelis, E. Boukas, L. Nalpantidis, and A. Gasteratos, "Stereo-based visual odometry for autonomous robot navigation," *International Journal of Advanced Robotic Systems*, vol. 13, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:62266727

[2] V. Umamaheswari, S. Amarjyoti, T. Bakshi, and A. Singh, "Steering angle estimation for autonomous vehicle navigation using hough and euclidean transform," in *Proceedings of the IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, 2015, pp. 1–5.

[3] A. rose Harman, "The environmental benefits of self-driving cars," https://greenerideal.com/news/vehicles/driverless-cars-environmental-benefits/, 2024, accessed: 2024-08-22.

[4] "Self-driving vehicles," https://www.government.nl/topics/mobility-public-transport-and-road-safety/self-driving-vehicles, Government of the Netherlands, 2024, accessed: 2024-08-22.

[5] About RDW. RDW. Accessed: 2024-08-22. [Online]. Available: https://www.selfdrivingchallenge.nl/about-rdw

[6] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022.

[7] evGrandPrix. Purdue University. [Online]. Available: https://engineering.purdue.edu/evGrandPrix/autonomous

[8] Formula Student Driverless. SAE International. Accessed: 2024-08-22. [Online]. Available: https://www.fsaeonline.com/

[9] Self Driving Challenge 2023. RDW. Accessed: 2024-08-22. [Online]. Available: https://www.selfdrivingchallenge.nl/previous-editions/edition-2023

[10] S. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 1,6, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:114862052

[11] E. Joa, Y. Sun, and F. Borrelli, “Monocular camera localization for automated vehicles using image retrieval,” 2021, arXiv preprint. [Online]. Available: https://doi.org/10.48550/arXiv.2109.06296

[12] W. Gates, G. Jati, M. Pratama, W. Jatmiko *et al.*, “A modest system of feature-based stereo visual odometry,” in *Proceedings of the IEEE 6th International Workshop on Big Data and Information Security (IWBIS)*, 2021, pp. 47–52.

[13] J. Sujatha *et al.*, “Computer vision based novel steering angle calculation for autonomous vehicles,” in *Proceedings of the IEEE 2nd International Conference on Robotic Computing (IRC)*, 2018, pp. 143–146.

[14] M. Aladem and S. A. Rawashdeh, “Lightweight visual odometry for autonomous mobile robots,” *Sensors*, vol. 18, no. 9, pp. 2837–2851, 2018.

[15] Y. Tanaka, A. Semmyo, Y. Nishida, S. Yasukawa, J. Ahn, and K. Ishii, “Evaluation of underwater vehicle’s self-localization based on visual odometry or sensor odometry,” in *Proceedings of the IEEE 14th Conference on Industrial and Information Systems (ICIIS)*, 2019, pp. 384–389.

[16] Z. Qiao, M. Zhou, T. Agarwal, Z. Zhuang, F. Jahncke, P.-J. Wang, J. Friedman, H. Lai, D. Sahu, T. Nagy *et al.*, “Av4ev: Open-source modular autonomous electric vehicle platform to make mobility research accessible,” 2023, arXiv preprint. [Online]. Available: https://doi.org/10.48550/arXiv.2312.00951

[17] A. Hernandez-Gutierrez, J. I. Nieto, T. A. Vidal-Calleja, and E. Nebot, “Large scale visual odometry using stereo vision,” in *Proceedings of the Australasian Conference on Robotics and Automation (ACRA)*, 2009.

[18] F. Sauerbeck, S. Huch, F. Fent, P. Karle, D. Kulmer, and J. Betz, “Learn to see fast: Lessons learned from autonomous racing on how to develop perception systems,” *IEEE Access*, vol. 11, pp. 44 034–44 050, 2023.

[19] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *Proceedings of the IEEE International conference on computer vision*, 2011, pp. 2564–2571.

[20] S. A. K. Tareen and Z. Saleem, “A comparative analysis of sift, surf, kaze, akaze, orb, and brisk,” in *Proceedings of the IEEE International conference on computing, mathematics and engineering technologies (iCoMET)*, 2018, pp. 1–10.

[21] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[22] Feature Matching. Accessed: 2024-08-22. [Online]. Available: https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html

[23] Feature extraction and matching. Accessed: 2024-08-22. [Online]. Available: https://pyflowopencv.readthedocs.io/en/latest/tutorial04.html

[24] Camera Calibration and 3D Reconstruction. Accessed: 2024-08-22. [Online]. Available: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html

[25] R. Arandjelovic and A. Zisserman, “All about vlad,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2013, pp. 1578–1585.

[26] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation,” in *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*, 2010, pp. 3304–3311.

[27] J. Á. B. Palma, M. N. I. Bonilla, and R. E. Grande, “Lane line detection computer vision system applied to a scale autonomos car: Automodelcar,” in *Proceedings of the IEEE 17th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2020, pp. 1–6.

[28] V. S. Dev, V. S. Variyar, and K. Soman, “Steering angle estimation for autonomous vehicle,” in *Proceedings of the IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 871–876.

[29] A. Howard, “Real-time stereo visual odometry for autonomous ground vehicles,” in *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, 2008, pp. 3946–3952.

[30] H. Halmaoui and A. Haqiq, “Feature detection and tracking for visual effects: Augmented reality and video stabilization,” in *Artificial Intelligence and Industrial Applications: Smart Operation Management*. Springer, 2021, pp. 291–311.

Study of Self-Driving Functionality: Lane Boundary Detection-Based Implementation in Small-Scale Karts

Raj Kumar Ashokan

***Abstract*—This paper presents the study carried out using lane boundary detection-based self driving functionality in small-scale karts using traditional approaches. This study was carried out as a follow-up study addressing the shortcomings of this approach implemented at the Self-Driving Challenge(SDC) 2023 edition. This implementation is adapted to be applicable to challenges pertaining to outdoor small-scale kart environments. A small-scale test kart set up is rebuilt including a simple CAN-communication network as in conventional cars. A complete vision-based autonomous pipeline is explored along with the suitable lateral control approaches based on the detected lane boundaries, for lane-center following. We also identify a significant discrepancy in a commonly used steer approach that claims to maintain the kart at the center of the lane. Unlike most other related works, real-time testing of the kart on a running track is demonstrated successfully under varying environmental conditions. Comprehensive discussion and extensive evaluation is done. Precision, Recall, and F1-scores of 0.97, 0.99 and 0.97 are achieved for the lane detection and a value of 1.395 pixels is achieved as a newly adapted Mean-Average-Perpendicular-Distance (MAPD) that indicate closeness to the ground truth. A mean deviation as low as 2-3 cm from the lane center towards the left is achieved throughout the runs. The algorithm achieves processing speeds of 30 FPS during the real-time autonomous runs. A supplementary video showcasing a demonstration of autonomous driving using the implemented pipeline can be found on https://tinyurl.com/5n7a9k3a**

***Index Terms*—self driving challenge, autonomous cars, small-scale karts, lane boundary detection, lateral control, steer estimation, lane center-following**

I. Introduction

Self-driving car technology has been making great strides towards becoming a reality and has been changing day to day lives in terms of road safety [1], ease in mobility, increased travel comfort, reduced emission and pollution levels [2], improved transport inter-connectivity etc. since its advent. Vehicles are increasingly equipped with advanced sensors, vision and control systems, providing them with autonomous capabilities [3] [4]. It has become inevitable to further probe into this field and research into the latest advancements in order to expand the knowledge in smart mobility.

In line with this, the Netherlands Vehicle Authority (RDW) has been organizing an annual competition, 'Self Driving Challenge (SDC)', since 2019. The RDW organizes this challenge with the futuristic goal of preparing itself for expanding its knowledge about autonomous vehicles, especially cars, and about the complex choices those vehicles make [5]. We participated in the SDC 2023 edition as a part of the team from the University of Twente. The main aim of this edition was to build a software stack to autonomously drive a lap as fast as possible at the specified track using an electric go-kart that is provided by the RDW.

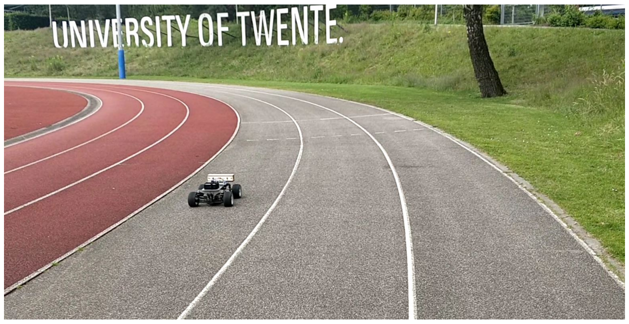


Fig. 1: Small-scale kart during an autonomous manoeuvre at the UTrack

Autonomous racing on karts provides a valuable testing field for algorithmic approaches related to autonomous driving. As this field is emerging and relatively new, a direct transfer of autonomous racing software to the autonomous passenger cars has not yet been accomplished [6]. However, increasingly, more autonomous racing challenges are organized using karts, such as the EV Grand Prix Autonomous Challenge [7] and Formula Student Driverless competitions [8]; these challenges induce valuable research that can be scaled up to passenger cars. Therefore, study of basic autonomous functionality behaviour using karts and small-scale karts can be a good starting point in the study of the same in cars, aligning with the motive of the SDC [5].

A total of six teams participated in the edition and our team secured the second prize. At the end of the challenge, our developed code was able to provide autonomous functionality to the provided go-kart but extensive testing was necessary in order to evaluate its autonomous behaviour. As the challenge was of a short stint, and as the kart was inaccessible after the challenge, in order to study this further, a small-scale kart, mimicking the basic functionalities of the SDC go-kart, was re-built at our university and the autonomous functionality was further developed, deployed and its behaviour was studied. The approaches we used for participating in the challenge is discussed in a separate preliminary paper included in the first part of this Master's thesis work. The follow-up comprehensive study using the small-scale kart was carried out as the second part of this Master's thesis work, which is primarily discussed in this paper. Overall, the objective of this Master's thesis work was to implement and study the behaviour of basic autonomous functionality in cars using karts and small-scale karts. Fig. 1 shows the small-scale kart during one of its autonomous manoeuvres.

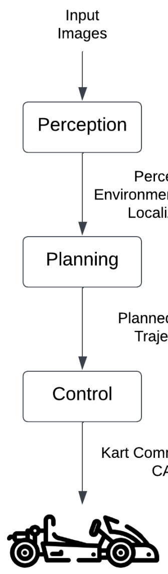

Input Images
Perception
Perceived Environmental Data & Localization
Planning
Planned Path / Trajectory
Control
Kart Commands via CAN

Fig. 2: Basic autonomous driving pipeline

The pipeline for a vehicle to drive itself autonomously from a point A to point B includes three major modules: perception, planning and control as in Fig. 2. Perception is the ability of a vehicle to perceive its surroundings and to know its own position in the environment using the data from its sensors. Planning is the process of generating the best path for the vehicle to traverse and reach its destination, based on the perceived environment taking into account the dynamic capabilities of the vehicle, presence of obstacles etc. Control is the process of converting the intended decisions into actions by sending commands to the actuators to obtain the desired movement [9].

In the SDC kart, cameras were meant to be the primary sensor setup, limiting the scope to visual perception based systems. Obstacle avoidance is also not considered in the scope of this work. One of the challenging perception tasks for an autonomous vehicle is estimating its current ego-pose or localization [10] [11]. Visual-based localization systems can be broadly based on traditional, learning or hybrid approaches. The state-of-the-art learning-based or hybrid approaches require a lot of data and processing power [6] [12] for considerable performance. However, during the SDC, owing to the limited processing capabilities of the kart, we resorted to using only traditional approaches.

Considering traditional approaches, Visual Odometry (VO), which is a process of estimating the translational and rotational movements of the camera using images, is often used for localization in autonomous vehicles [13] [14]. At the SDC, we attempted a monocular visual odometry-based approach due to the lack of a stereo setup. Relative localization was obtained using this approach but integrating the absolute scale could not be successfully performed due to lack of any other additional information. Therefore, in the later part of the challenge, we adopted a lane boundary detection-based road following approach, which is another commonly used approach; we successfully utilized this approach at the SDC to make the kart autonomously drive a distance of 1 km. However, the exhibited behaviours of the kart could not be studied including significant oscillations of the kart, unintended deviations from the center of the lane etc., highlighting these as the shortcomings at the challenge. Despite the shortcomings, this approach has potential to work reliably. Thus, in the follow-up study, this approach was adapted and used to study the autonomous functionality behaviour, aligning with the primary aim of this work. More details about the approaches used at the challenge, the results obtained and the discussions of our work can be found in the preliminary paper included in the first part of this Master's thesis work.

In order to serve as the test platform for the follow-up study, re-build of a small-scale kart was carried out incorporating the same basic functionalities as the SDC kart. The small-scale or reduced scale vehicles are normally derived from RC cars with modified additional hardware and are developed mainly for the purpose of testing autonomous software. They can be comparable to racing go-karts as they reach high speeds and accelerations for their size [6]. The autonomous functionality implementation was adapted considering the limitations of the small-scale kart environment. Especially, the low-camera heights introduce more challenges and the suitability of the techniques and the overall process were investigated.

Nevertheless, the basic steps involved in the modules remained the same. The perception module includes feature extraction, detection of lane boundary markers, and lateral localization relative to the lane markings. In the planning module, steer angle is calculated in order to generate the trajectory for lane following, i.e., for the kart to be positioned at the center of the lane. The lateral control module executes the movements, including error handling process, to maintain the planned path.

Finally, extensive real-time tests and quantitative evaluation were carried out to analyze the behaviour. Understanding of the behaviour of the runs carried out at the SDC challenge was a secondary outcome of this study. A methodological flaw was also identified in an existing steer computation technique that claims to keep the vehicle in the center of the lane, adding to the novel contributions of this study.

Therefore, this research and analysis revolves around the implementation of vision-only-based lane boundary detection and lateral control in small-scale karts, especially relevant to low camera heights and for positioning the kart at the center of the lane. Building on the adaptations from the SDC implementation, we primarily focus on the following research questions:

1) How can existing lane boundary detection algorithms be optimized for use with small-scale karts?
2) How can lateral control be applied using the detected lane boundaries for directing a small-scale kart's movement?

3) How can the derived steering angle be quantitatively evaluated?
4) Can utilizing the input feed from multi-cameras impact the behaviour of the autonomous functionality?

The remainder of this work is organized as follows: Section II reviews the related literature with a focus on the research questions. Section III outlines the re-build of the small-scale kart. Section IV provides an overview of the data collection process. Section V describes in detail the methods applied in this study. Section VI presents the achieved results and discusses the findings. Section VII summarizes the findings of the study and concludes the work with scope for future enhancements.

II. Related Works

Relevant to the RQs, we have summarized our findings of the reviewed work in three categories: vision-based lane boundary detection, steer calculation methods based on the detected boundaries to keep the kart in the lane center, and evaluation of the generated real-time steer values. Fewer works specifically focus on small-scale karts and go-karts; therefore, our search is confined not only to karts but also includes cars, as both are comparable to an extent. Furthermore, the discussions here are limited to classical approaches and do not include learning based approaches.

Work by Uma et al [2] employs a Euclidean distance and trigonometry-based approach for boundary detection and steer computation. Alternatively, in cases of non-linear, uneven contrast, and irregularly intense input data it uses SHT-based and lane-midpoint-based methods respectively. The work claims to achieve successful testing in collected real-time day and night video samples including cases of shadow and irregular contrast, but no real time testing is shown. For evaluation, it mentions visual analysis, but misses to discuss any validation criteria or quantitative evaluation.

Dev et al [15] utilize a series of pre-processing steps followed by Canny edge detection and SHT for lane boundary detection. Steer value is computed based on the vanishing point. They claim that their work performs well in terms of accuracy, computation time, and fault detections, but utilize only visual evaluation based on collected real-time day and night video samples, without mentioning any validation criteria or metrics.

Kittithuch et al [16] use a small indoor RPi robot and explore a similar boundary detection approach as [2] [15]. However, they compute an initial reference of the slope and angles of the lane boundaries; then they detect only either boundary to compare its parameters with this reference to calculate the steer. On a indoor test setup, navigation of the kart within the lane is reported, although with a lateral swinging behaviour. Lane boundary or steer evaluations are not shown but the kart's traversing behaviour is plotted along with the offset.

In the work by Ranjith et al [12], Gaussian Mixture Model and Expectation Maximization algorithm is used to extract the lane boundaries followed by a vanishing point based steer computation. Simulation testing is performed and steering evaluation is carried out on the Udacity dataset resulting in a high RMSE. Again, no real-time testing is shown on hardware. Mahersatillah et al [17] also propose a segmentation based boundary detection to check the offset of the car from the lane's center, only to produce a lane departure warning but not a corrective steering; it also works on the assumption that the car can maintain its position around the center.

Karouach and Ivanov [18] present their work on lane detection and lane following by adapting a miniature indoor car for the Carlo Cup competition. Their improved algorithm focuses on the lane detection part using contours and utilizes the vanishing point to keep the 1:10 car within the lane. However their algorithm was tested under controlled settings of the competition and not under outdoor or different settings.

Khanh et al [19] explore autonomous navigation in small and narrow indoor environments using a robot prototype. The conventional approach used in their work utilizes only the near field regions necessary for the lane detection and applies Inverse Perspective Mapping for easier processing. The work then employs Canny edge detection and SHT for boundary detection and uses linear average of the detected lines for steer computation. Testing is carried out on a 2m indoor setup with controlled lighting conditions, and also on a Udacity dataset of bright sunny images in outdoor environments. Steer evaluation using MAE and R-squared and computation time is also discussed.

In the work by Mohamed et al [20], again the standard edge detection and SHT techniques are applied, using Inverse Perspective Mapping. They consider the lateral and the yaw error for computing the steer angle and minimize both individually using implementation of controllers respectively. However, they do not perform any real-time testing and rather work with simulation images, for which statistical analysis of steer value and lateral offset is presented. In Jose et al [21], similarly, Inverse Perspective Mapping view is used. They use histogram distribution of white areas followed by point grouping for lane detection and then fitting of polynomial regression for lane boundary detection, and steer is computed using road curvature. They perform indoor tests and present the RMSE and MAE for line detection.

Summing up, though more works are focused on the lane boundary detection, the relevancy to outdoor small-scale karts is limited; even the fewer works that are relevant are explored in controlled optimal conditions with less noise. Therefore, the suitability to outdoor noisy environments as in our case is to be investigated. Concerning steer computation, excluding IPM techniques, four of the works claim to maintain the kart running in the lane center. These rely on vanishing point and intercept-angle relationship techniques. We utilize these two and investigate the suitability. Moving on to real-time testing and evaluation, this seems to be the least addressed topic. Most of the works state only visual evaluation of the lane boundary detection, derived steer, and performance of the run but fail to discuss any evaluation criteria or quantitative analysis.

Testing is mostly carried out using collected data or sim-

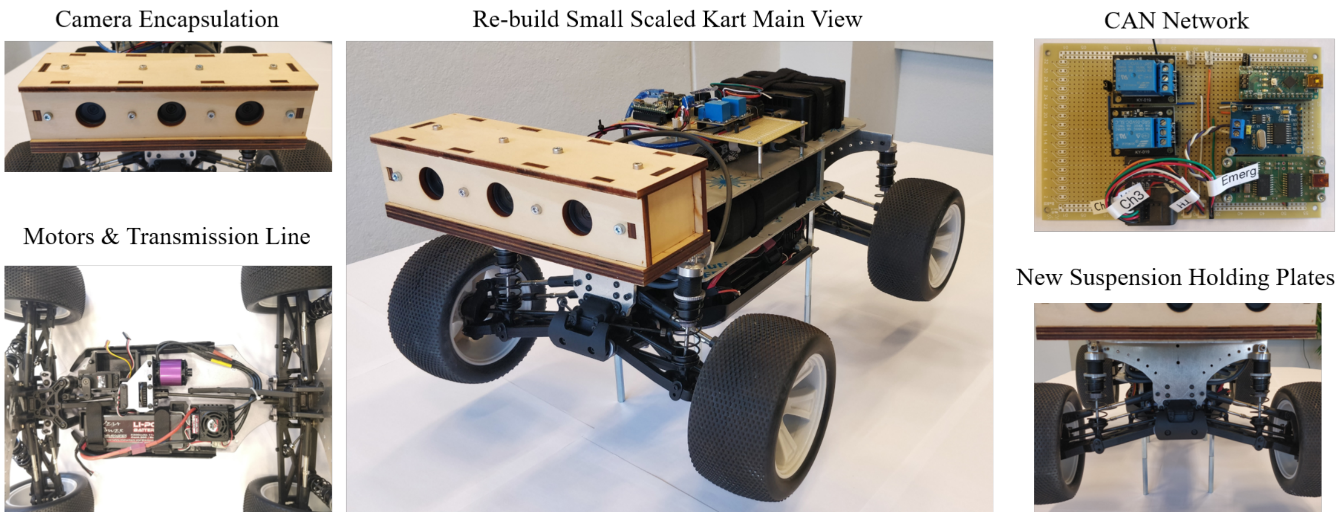

Camera Encapsulation
Re-build Small Scaled Kart Main View
CAN Network
Motors & Transmission Line
New Suspension Holding Plates

Fig. 3: Rebuilt small-scale kart

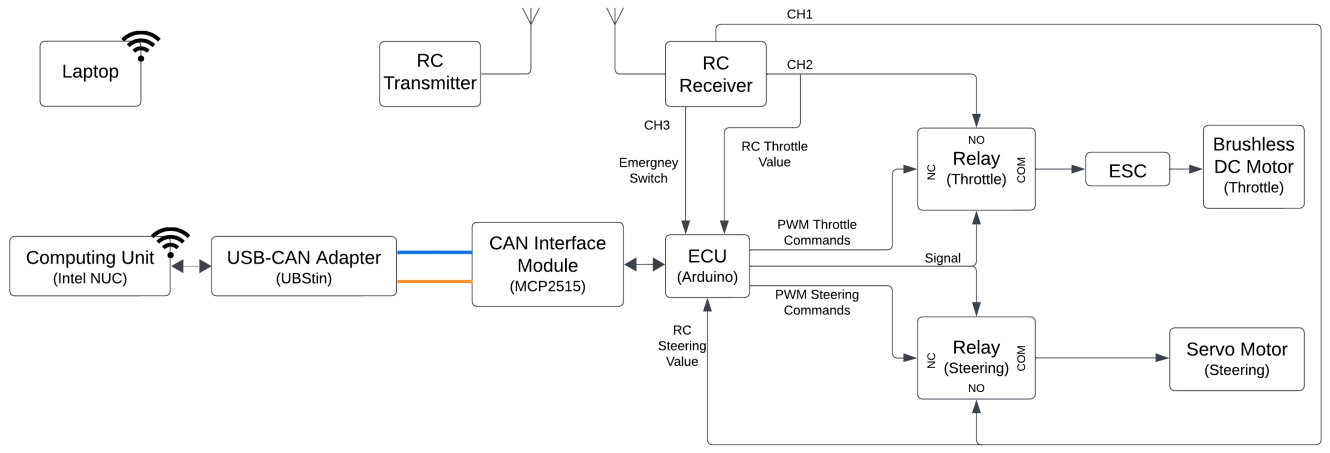


Fig. 4: Block diagram of the communication network of the kart

ulation dataset or on indoor (miniature) set ups. This cannot effectively translate to real world testing as the real-world environment is complex and deals with differing conditions, unforeseen scenarios, hardware compatibility issues, and performance constraints. Therefore, we focus on real-world testing using a rebuilt small-scale kart along with quantitative evaluation of the detected lane boundaries, computed steer and its reliability.

III. Re-build of Small-Scale Kart

We considered a 1:8 scale RC HIMOTO kart for the rebuild framework. For better control, the original brushless DC motor was replaced with a new one to reduce the speed from 80 km/hr to approximately 23 km/hr. Servo motor for steering and electronic speed control for braking is used. A Four-Wheel-Drive (4WD) mechanism drives the kart. Separated power supplies for the motors and the computing unit were used. Intel i5 Processor 16GB RAM was chosen, which is the same as the one used in the SDC kart. Unlike the RDW kart, here, only a single ECU controls both the throttle ESC and the steer servo. A CAN based communication network was built connecting the computing unit and the ECU(s) via

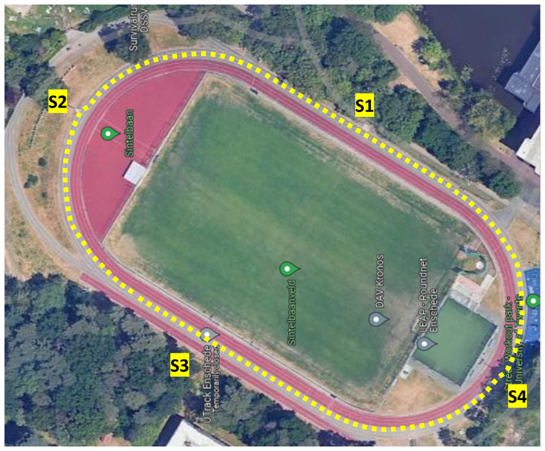


Fig. 5: UTrack with four segments used for autonomous driving

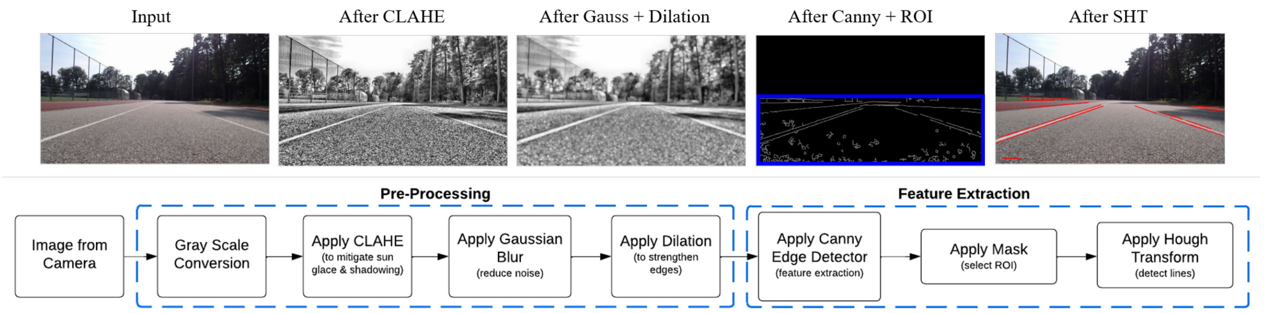

Input
After CLAHE
After Gauss + Dilation
After Canny + ROI
After SHT
Pre-Processing
Feature Extraction
Image from Camera
Gray Scale Conversion
Apply CLAHE (to mitigate sun glace & shadowing)
Apply Gaussian Blur (reduce noise)
Apply Dilation (to strengthen edges)
Apply Canny Edge Detector (feature extraction)
Apply Mask (select ROI)
Apply Hough Transform (detect lines)

Fig. 6: Perception module with focus on pre-processing and feature extraction steps illustrated using sample outputs

CAN bus trans-receiver interfaces, enabling the units to both listen for inputs from and to send feedback on the CAN bus. The kart was made controllable by using either RC mode (2.4GHz) or program control mode (either autonomous or keyboard commands). Programs on the computing unit are executed by a ground station (typically a laptop or a mobile device) connected via a mobile hotspot network. Notably, an emergency stop mechanism that can be activated by using a RC channel was also incorporated in the communication network, considering safety aspects.

The algorithm that is programmed in the ECU decodes and filters the received CAN messages based on the message identifiers and data payloads, to execute a specific task (including any of the above functionalities). The chassis was remodeled with newly designed suspension plates to increase load capacity and to accommodate two platforms, one for the power supply and the communication circuit and the other for the cameras and the computing unit. Initially left, center, right cameras were mounted directly on the top platform. However, this resulted in image instabilities due to lack of rigidity. Hence, an encapsulation setup was designed to house the cameras and this setup was attached to the platform with the center of the camera at a height of 19 cm from the ground. The final rebuilt kart and the block diagram of the entire communication circuit which was built incorporating all these mechanisms are as shown in Fig. 3 and Fig. 4 respectively.

IV. Data Collection

The UTrack athletic track, at the University of Twente, consisting of multiple lanes of width 120cm and length 400m was chosen as the test field. As wide as a normal road is for a full-sized kart, similarly proportioned is each lane of the UTrack for our small-scale kart with low camera height. The track is in the form of a loop with two straight segments(S1,S3) and two semicircle segments(S2,S4), as shown in Fig. 5. Data collection was performed at this track in real-time by running the kart manually using RC and keyboard command modes in both clockwise (S1,S4,S3,S2) and anticlockwise (S1,S2,S3,S4) directions. Images were captured at a resolution of 848x480 in different natural lighting conditions and seasons. Additionally, data was recorded on both the asphalt and synthetic lanes (red-colored lanes) of the UTrack, and also by driving the kart in opposite directions and zig-zag manoeuvres and in varying speed modes. Similar to the SDC kart, CSV files containing throttle and steer data were additionally recorded using the concept of multi-threading.

V. Methodologies

Aligning with our primary aim of evaluating the basic autonomous functionality that was implemented already during the SDC, re-usability of the code is primarily considered here, maintaining the basic pipeline. Only the adaptations that were made to the code for it to work on the small-scale kart are discussed here.

1) Pre-Processing and Feature Extraction: In the perception module, in order to accommodate for the changes in the vision system arrangement and the road dimensions, tuning all the parameters in the lane detection step was attempted by utilizing the center camera data subsets. However, issues like sun glare, shadow effects, noisy textures, and weak edges were encountered. These were not mostly faced during the challenge due to the favourable conditions that comes along with racetrack environments, such as absence of trees or shadows and the comparatively higher placement of cameras. Consequently, a series of pre-processing steps were necessary after initial grayscale conversion. The applied steps are as shown in the block diagram in Fig. 6. Contrast Limited Adaptive Histogram Equalization (CLAHE), which is a technique that enhances image contrast by applying histogram equalization to small regions while limiting noise amplification, was used to mitigate the shadow and sun glare effects. This was followed by Gaussian blur to reduce noise and then a dilation operation to strengthen the weak edges. After these pre-processing steps, the feature extraction steps (Canny edge detector, Region of Interest selection, Standard Hough Transforn) were performed to extract the possible lines.

2) Feature Filtering and Localization: The same set of line filtering and refinement steps as used in the SDC implementation were applied next with the following modifications. Preponing the process of classifying the potential lines into left and right line categories based on the sign of the slope was performed; this simplifies the process and helps in choosing better and robust threshold parameters. Positional filtering was applied next followed by slope and intercept thresholding. An important issue encountered in these steps in the SDC implementation was that narrow threshold ranges

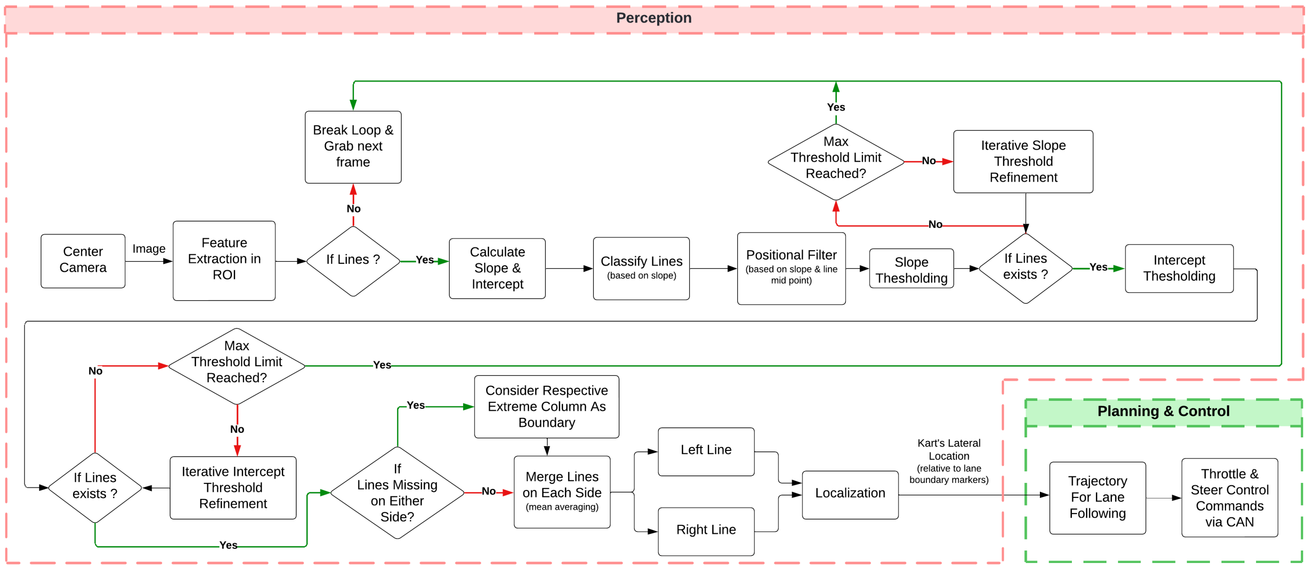
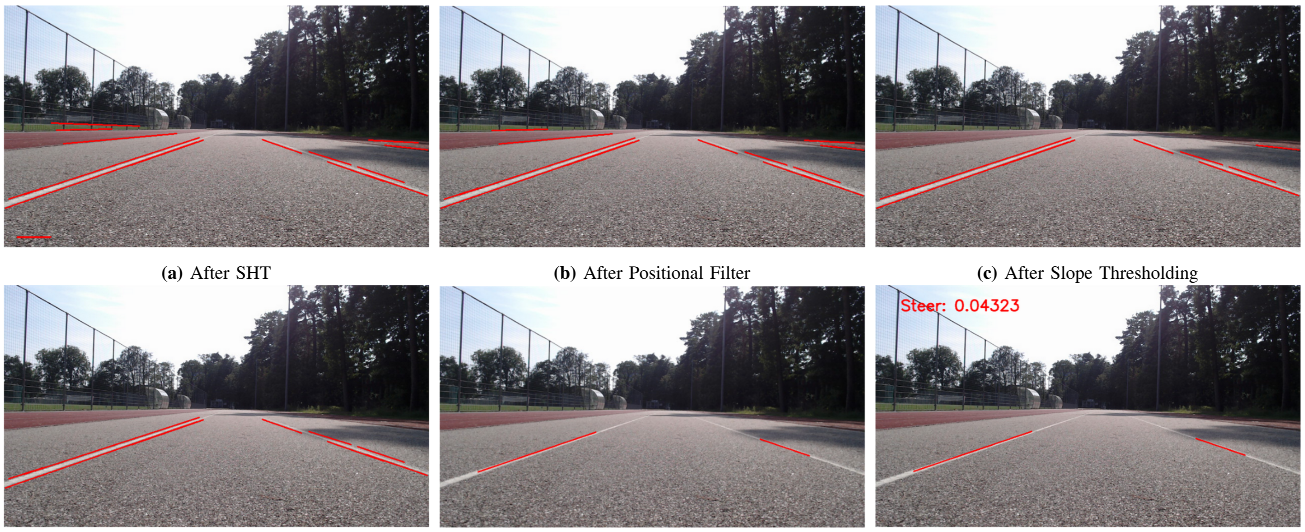

Perception
Center Camera
Image
Feature Extraction in ROI
If Lines ?
No
Yes
Break Loop & Grab next frame
Calculate Slope & Intercept
Classify Lines (based on slope)
Positional Filter (based on slope & line mid point)
Slope Thesholding
If Lines exists ?
Yes
Intercept Thesholding
Max Threshold Limit Reached?
Yes
No
Iterative Slope Threshold Refinement
No
Max Threshold Limit Reached?
No
Yes
No
Iterative Intercept Threshold Refinement
If Lines exists ?
Yes
If Lines Missing on Either Side?
Yes
No
Consider Respective Extreme Column As Boundary
Merge Lines on Each Side (mean averaging)
Left Line
Right Line
Localization
Kart's Lateral Location (relative to lane boundary markers)
Planning & Control
Trajectory For Lane Following
Throttle & Steer Control Commands via CAN

Fig. 7: Perception module with focus on filtering steps



(a) After SHT **(b)** After Positional Filter **(c)** After Slope Thresholding

(d) After Intercept Thresholding **(e)** After Clustering **(f)** After Merge & Steer Calculation

Fig. 8: Sample outputs at intermediate steps (cont. from Fig. 6)

could not be chosen for these values; narrow ranges resulted in valid lines being filtered out in the images belonging to different parts of the track as the track width varied. On the other hand, having a broader range resulted in irrelevant lines still being present in the image.

In order to resolve this, we performed an iterative threshold refinement process. For this, we start with a narrow range of threshold that suits major parts of the track and try applying this for the remaining parts of the track. If no lines are detected, the range is slowly increased until a maximum value, to check for lines. If no lines are still found, the frame is dropped and the algorithm grabs the next image. Whereas, if lines are found, intercept thresholding is tuned next, using the same iterative refinement method, until a maximum value, after which the frame is dropped and the entire process re-begins by fetching the next frame. The rest of the steps to result in the final boundary lines remained the same. The overview of the perception module with a focus on the filtering steps is as shown in Fig. 7.

3) Lane Following: Next, in the planning module, the steer computation was the same as used in the SDC implementation, intended to keep the kart at the center. However, owing to results analyzed during the real-time tests (as discussed later in the Results section), we decided that this is not the appropriate method for the intended purpose. Subsequently, we opted for a different method based on adaptations to the work in [2]. The ratio of the y-intercepts of the obtained lines is considered for this purpose. As can be seen in the image in Fig. 9, y_{left} and y_{right} denote the intercept values of the obtained lines with the y-axis; x_{left}

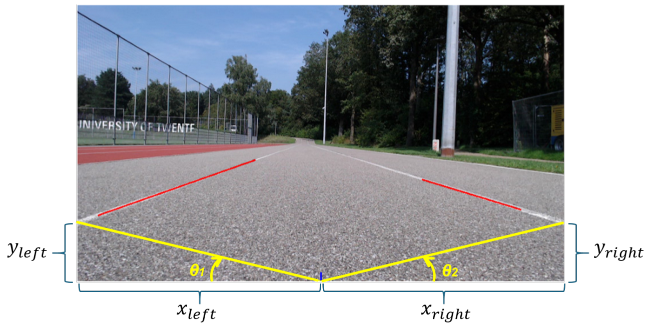


Fig. 9: Notations used for steer angle calculation. Here, y_{left}, y_{right}, x_{left}, x_{right} are intercept values the lane boundary lines make with image x and y axis, and θ_1, θ_2 denotes the deviations of the boundary lines from the positions corresponding to the center of lane

and x_{right} are pixel distances equalling half the image width. These values are computed and based on that, the angles θ_1 and θ_2 representing the deviations of the boundary lines, from the ideal positions corresponding to the center of the lane, are obtained. The steer value (ϕ) is thus calculated as:

$$\phi = \begin{cases} \dfrac{\theta_2 - \theta_1}{S_{max}} & \text{, when both lane boundaries are detected} \\ \\ \dfrac{\theta_1 - \theta_2}{S_{max}} & \text{, when only one lane boundary is detected} \end{cases} \tag{1}$$

The steer directions based on the possible θ_1 and θ_2 values are represented in Table I.

Theta Values ($\boldsymbol{\theta}$)	**Steer Direction**		
	When both lane boundaries visible $\boldsymbol{\phi = \theta_2 - \theta_1}$	**When only right lane boundary visible ($\boldsymbol{\theta_1 = 0}$) $\boldsymbol{\phi = \theta_1 - \theta_2}$**	**When only left lane boundary visible ($\boldsymbol{\theta_2 = 0}$) $\boldsymbol{\phi = \theta_1 - \theta_2}$**
$\theta_1 > \theta_2$	Left	Left	Right
$\theta_1 < \theta_2$	Right	Left	Right
$\theta_1 = \theta_2$	No Steer	No Steer	No Steer

TABLE I: Steer directions based on θ_1 and θ_2 values

4) Off-Track Avoidance: Off-track avoidance module utilizing the left and right camera images (as designed in the SDC implementation) is incorporated here with adaptations. For both the left and right camera feed, the Region of Search (RoS) was redefined after analyses. The parameters for the line detection steps were tuned. Using these tuned parameters, line detection was performed. After this, no further filtering steps were necessary in the SDC implementation. On the contrary, here, owing to the very close proximity of the cameras to the road, more noise was included; this necessitated the use of line filtering steps to obtain the final detected line. The flow of the processes in the off-track avoidance module is as shown in Fig. 10. For the proportional computation of the steer value, as can be seen in Fig. 11, this detected line's x-intercept with the image was considered

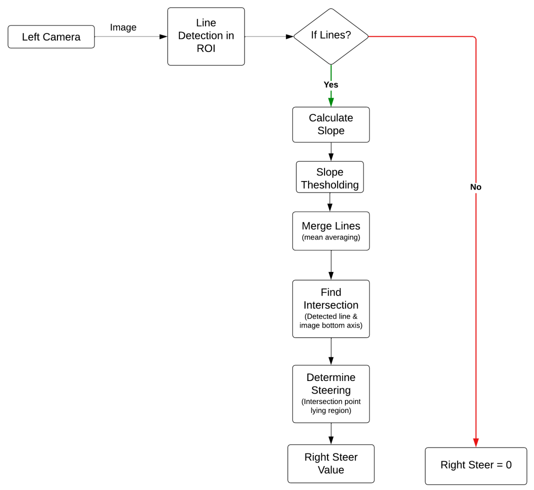
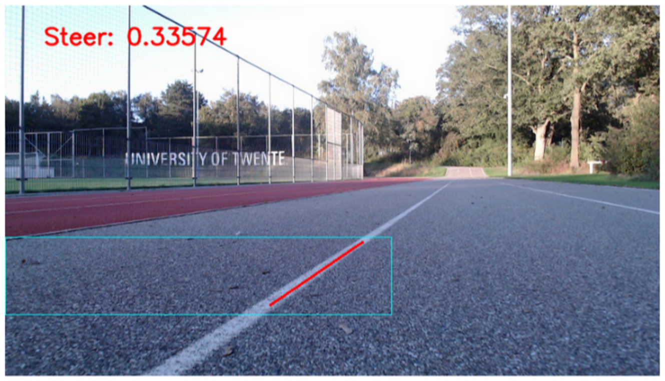
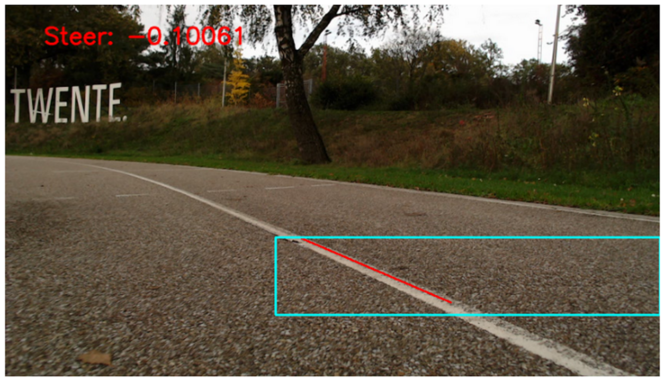

Left Camera
Image
Line Detection in ROI
If Lines?
Yes
No
Calculate Slope
Slope Thesholding
Merge Lines (mean averaging)
Find Intersection (Detected line & image bottom axis)
Determine Steering (Intersection point lying region)
Right Steer Value
Right Steer = 0

Fig. 10: Off-track avoidance module utilizing left camera

rather than considering the intersection of the line with the RoS boundary. This increases stability.



(a) From left camera



(b) From right camera

Fig. 11: Sample outputs from off-track avoidance module

The concept of uni-modal fusion is also followed here, combining the off-track avoidance in the decision-making module. The overview of this module is as shown in Fig. 12. Consequently, the final steer values are passed as control commands via the CAN network to the actuators.

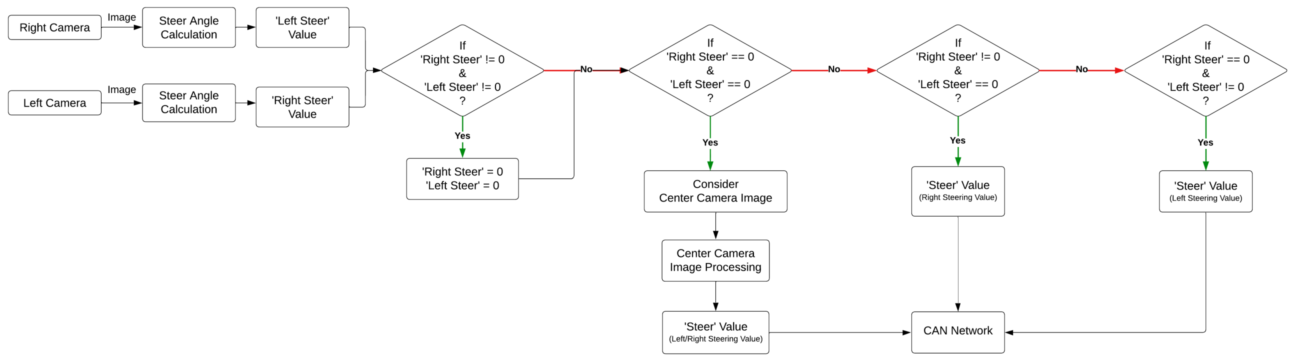

Right Camera
Image
Steer Angle Calculation
'Left Steer' Value
Left Camera
Image
Steer Angle Calculation
'Right Steer' Value
If 'Right Steer' != 0 & 'Left Steer' != 0 ?
Yes
'Right Steer' = 0 'Left Steer' = 0
No
If 'Right Steer' == 0 & 'Left Steer' == 0 ?
Yes
Consider Center Camera Image
Center Camera Image Processing
'Steer' Value (Left/Right Steering Value)
No
If 'Right Steer' != 0 & 'Left Steer' == 0 ?
Yes
'Steer' Value (Right Steering Value)
No
If 'Right Steer' == 0 & 'Left Steer' != 0 ?
Yes
'Steer' Value (Left Steering Value)
CAN Network

Fig. 12: Decision-making module involving uni-modal fusion

A. *Evaluation Metrics*

In line with the research motive, evaluation and metrics are discussed for the detected lane boundaries, the derived steer values and the kart's behaviour in real-time.

1) Lane boundaries: The detected lines are initially evaluated by assessing the closeness to the ground truth. The ground truth and detected lines can be non-parallel, non-overlapping and of different lengths. Because of this, standard metrics like MAE, MSE and RMSE are not directly applicable. Therefore, we use a new metric, 'Mean Absolute Perpendicular Distance (MAPD)', by adapting the standard MAE to the context of lane detection. If $Ax + By + C$ and $Mp+Nq+O$ represents the ground truth (G) and the detected (D) line equations respectively, then:

$$MAPD_{G\to D} = \frac{1}{2a}\sum_{k=1}^{a}\sum_{i=1,2}\frac{|M_k x_{k,i} + N_k y_{k,i} + O|}{\sqrt{M_k^2 + N_k^2}} \quad (2)$$

$$MAPD_{D\to G} = \frac{1}{2a}\sum_{l=1}^{a}\sum_{j=1,2}\frac{|A_l p_{l,j} + B_l q_{l,j} + C|}{\sqrt{A_l^2 + B_l^2}} \quad (3)$$

$$MAPD = \frac{MAPD_{G\to D} + MAPD_{D\to G}}{2} \quad (4)$$

where (x_i,y_i) and (p_j,q_j) denote the end point coordinates of the ground truth and the detected line, a is the number of images considered. Computing for both $G \to D$ and $D \to G$ ensures robustness against asymmetries. Based on the $MAPD$ values that are separately calculated for the left and right lines, a threshold based criteria classifies the lines into a true or a false detection. We use a confusion matrix to sum this up for all the images, providing a comprehensive view which can be useful to pinpoint specific type of errors.

2) Derived steering angle: The steer values recorded from manual driving cannot be considered as the ideal ground truth values to follow a known path, as manual driving is prone to errors and imperfections, and the control can vary widely depending on the driver. Rather, an appropriate way is to create a model of the vehicle and to define its motion. A vehicle can be modeled using either kinematic or dynamic modelling. We consider the kinematic bicycle model that is typically used in low-speed scenarios. This

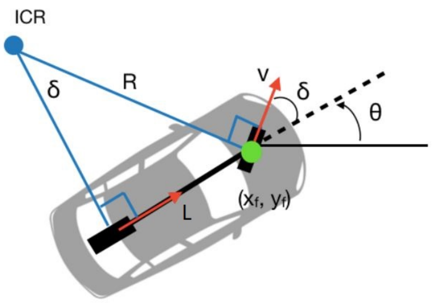


Fig. 13: Front axle kinematic bicycle model [22]. Here ICR is instantaneous center of rotation, R is radius of curvature, L is length of kart (between wheel centers), θ is heading angle, δ is desired steer and (x_f, y_f) is reference point taken at the front axle center

model is a simplified representation of a vehicle's motion that approximates the vehicle with a single front and rear wheel per axle and focuses on the geometric aspects [23], as shown in Fig. 13. Here ICR is instantaneous center of rotation, R is radius of curvature, L is length of kart (between wheel centers), θ is heading angle, δ is desired steer and the reference point is taken at the front axle center. The ideal steer value can thus be computed as:

$$\delta = \sin^{-1}(\frac{L}{R}) \quad (5)$$

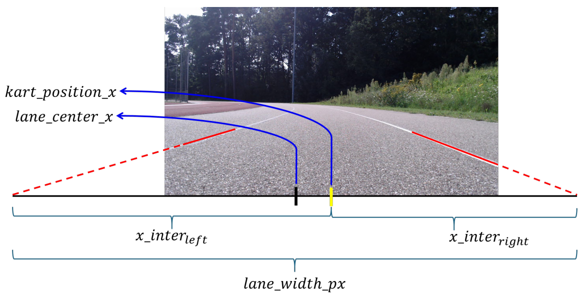


Fig. 14: Notations used for calculating the deviation of the kart

3) Behaviour of the kart: In order to analyze how well the kart maintains itself at the center of the lane during the run, the offset of the kart from the center of the lane needs to be evaluated. For this we utilize the detected lane boundaries in the image. As can be seen in the Fig. 14, we compute the lane width in pixels and the center of the lane in pixels and utilize these to calculate the deviation in centimetres as given in the below set of equations:

$$lane\_center\_x = \frac{x\_inter_{left} + x\_inter_{right}}{2} \quad (6)$$

$$lane\_width\_px = |x\_inter_{left}| + |x\_inter_{right}| \quad (7)$$

$$px\_cm\_convert = \frac{lane\_width\_cm}{lane\_width\_px} \quad (8)$$

$$deviation\_px = kart\_position\_x - lane\_center\_x \quad (9)$$

where $kart\_position\_x$ is the bottom center of the image and denotes the current position of the car.

$$deviation\_cm = deviation\_px \times px\_cm\_convert \quad (10)$$

The mean deviation in centimeters for an entire run can be given by:

$$MAE_{deviation} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \overline{y_i}| \quad (11)$$

where y_i is the ideal deviation/offset, $\overline{y_i}$ is the actual deviation/offset from the center of the lane and n is the number of images considered.

VI. Results and Discussions

The outcome of the real-time tests at the UTrack are discussed here initially using the first steer method followed by the new one. Using the first method, while excluding the off-track avoidance module, the kart ran straight autonomously in the straight segments but exhibited a strong oscillating behaviour around a boundary, throughout the turns, exiting and entering back the lane repeatedly. Such behaviour was observed in both the semi-circle segments, in all the runs. When including the off-track module, the same behaviour was observed except that the oscillations stayed within the lane space and the kart didn't cross the boundaries, proving the impact of the off-track module. On analyzing, this pattern was attributed to the following reasons:

i) Primarily, we found that this steer approach is capable of handling only the heading angle error and not the cross-track (lateral) error as can be seen in Fig. 15. This enabled a continuous excessive unidirectional steer in the direction of the turn making the kart go out of the lane.

ii) Mismatch between the periodic CAN sending speed and the algorithm processing speed resulting in delayed messages or unintentional repeat of steer values.

iii) As the kart exited the lane, making only one boundary visible, the extreme column of the image considered in place of the missing boundary made the kart to enter back into the lane.

As both the heading and lateral errors need to be addressed for the kart to be at the center of the lane, we inferred that this steer approach is not suitable for the purpose, even in straight segments; in these segments, the runs performed with near-edge start positions showed that the kart maintained the same straight offset, though without wobbling, proving our inference. To address both the heading and lateral errors, this steer method was replaced by a different one as in Eq. 1. Additionally, tuning of CAN sending speed as close as possible to the processing speed was performed.

On testing this new method with and without the off-track module, the kart was observed to run autonomously at the lane center throughout the track, forming a complete loop, without any oscillating behaviour. The same was achieved in multiple lanes, in both directions and with different starting positions and locations. Hence the off-track module was not seemingly utilized as the center camera did not fail in any of the places.

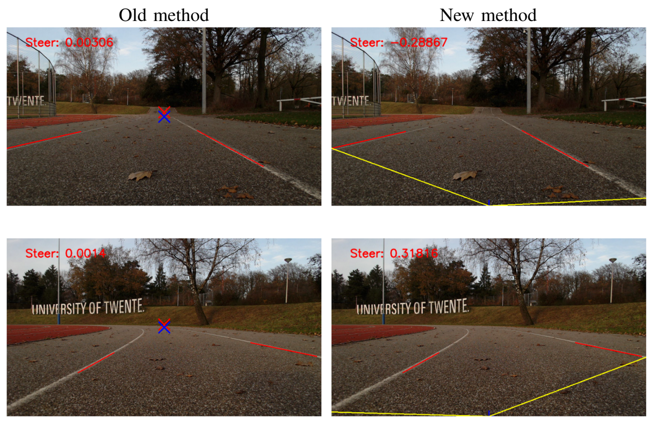


Fig. 15: Output comparison between different steer approach in S1(first row) and S2(second row) segments of the track

TABLE II: Confusion Matrix - Lane Detection

	Predicted Positive	**Predicted Negative**
Actual Positive	476	4
Actual Negative	10	470

1) Lane boundaries: In terms of evaluation, for lane boundary detection, the resulting confusion matrix is as shown in Table II. For calculating this confusion matrix, a dataset of 960 images was considered. This included images collected from both manual and autonomous driving. Since this evaluation was intended to assess only the lane boundary detection and not the autonomous driving, we considered images collected from manual driving as well, in order to include images from inconsistent driving scenarios. In addition to this, the dataset also contained images from clockwise and anticlockwise drives, so as to incorporate both left and right turns. The images spanned different lighting conditions taken at different times of the day and seasons, making the dataset diverse.

Out of the 960, the dataset included 480 images where both the boundary lines were present (with ground truth label as ”Lines Present”) and 480 images where both the boundary lines were absent (with ground truth label as ”Lines Not Present”). An image selection script was used to choose the images in a distributed manner for the evaluation dataset. This ensures a balanced dataset and forbids bias. The images were selected from the runs that fall under the operating domain conditions of the implemented autonomous functionality i.e., the scenarios that the algorithm was not designed to handle were not included.

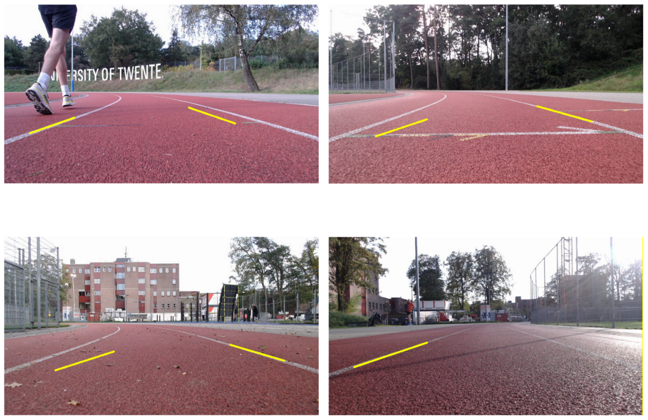


Fig. 16: Sample outputs showing false negatives (FN) represented as yellow lines

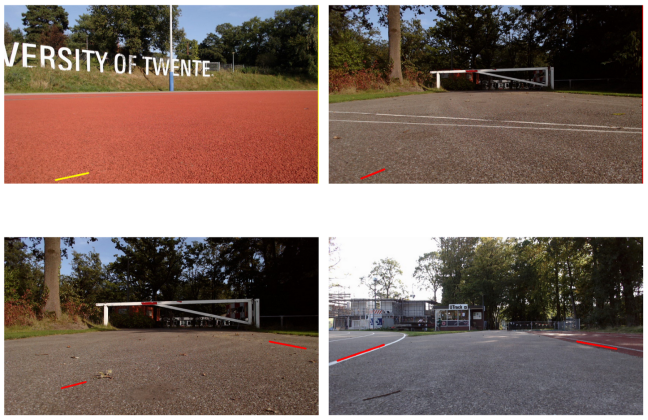
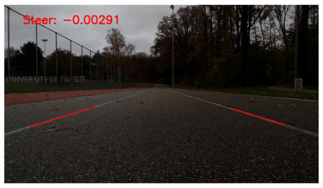
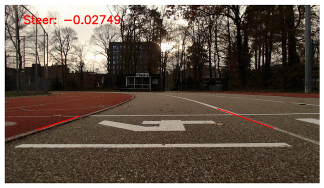


Fig. 17: Sample outputs showing false positives (FP) represented as yellow and red lines

For classifying a detected line in an image into a True Positive (TP) or a False Negative (FN), a pre-defined range of MAPD values was set as the threshold range. Detected lines that also have an MAPD value within this range would be considered a TP, whereas, a detected line with a MAPD value outside this range would be still considered a FN. A line is classified as False Positive (FP) when it is actually not present in the image but is detected by the algorithm. Accordingly, an image was classified as TP when both the detected lines were correctly classified under the TP category, and an image with no lines was classified as True Negative (TN) when no lines were detected in the image. An image was categorized as FP when either or both the lines was classified as FP and similarly for a FN category. Sample image outputs showcasing scenarios of lines classified as FP and FN are as shown in Fig. 16 and Fig. 17. The images in the entire dataset were classified accordingly resulting in the shown confusion matrix. The metrics Precision, Recall, and F1-score were computed from the confusion matrix. In our context, Precision gives the proportion of the correctly identified lane lines among all the detected lines and this is important to reduce the False Positives that occur, as it may lead to sudden unintended steer values in the wrong direction. On the other hand, Recall gives the proportion of the actual lines that were correctly identified. This is crucial in avoiding False Negatives. F1-score considers a harmonic mean of Precision and Recall to provide a balanced measure when both the FPs and FNs need to be minimized. Precision, Recall and F1-score were computed to be 0.97, 0.99 and 0.97 respectively on the chosen balanced dataset. This indicates that the algorithm is good at reducing ”False” scenarios increasing the reliability of lane line detections.

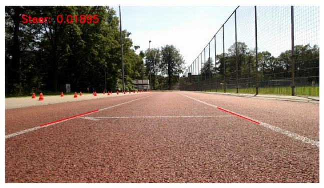


(a) Low light

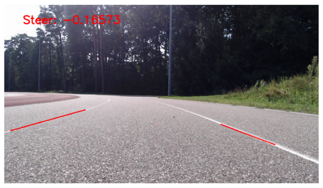


(b) Other lane markers

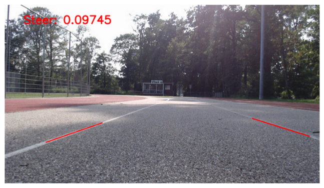
(c) Rainy days



(d) Synthetic lane



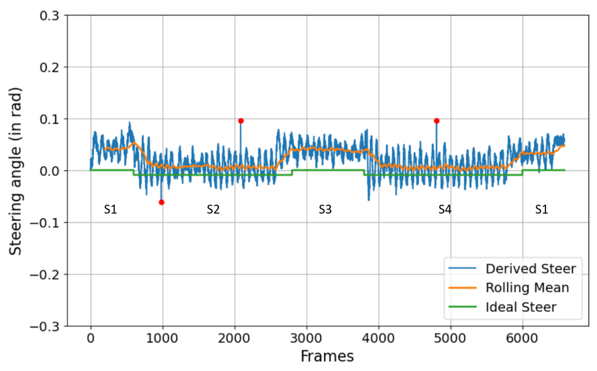
(e) Sun glare & road reflection



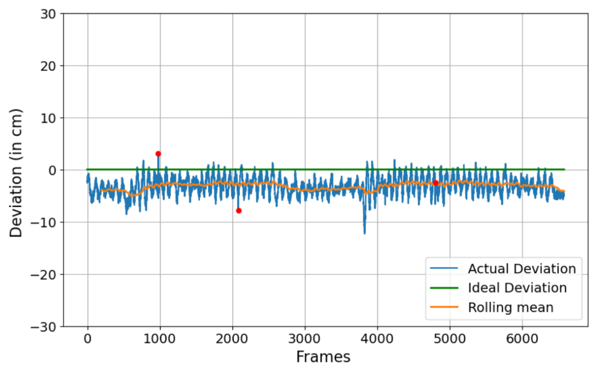
(f) Tree shadow & weak left lane boundary

Fig. 18: Sample outputs for other challenging scenarios

In the images categorized as True Positive, MAPD was computed for the left and the right lane lines separately. In our considered evaluation dataset, for the left lines, the $MAPD_{left}$ was obtained as 1.450 pixels with a variance of 0.779 and a standard deviation of 0.853 pixels; this indicates the closeness of the detected left lines to the ground truth in the entire dataset. Similarly, for the right lane lines, the obtained $MAPD_{right}$ value was 1.340 pixels with a variance of 0.588 and standard deviation of 0.745 pixels. This resulted

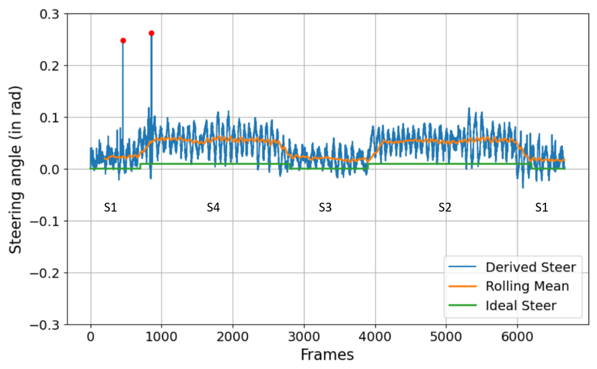


(a)

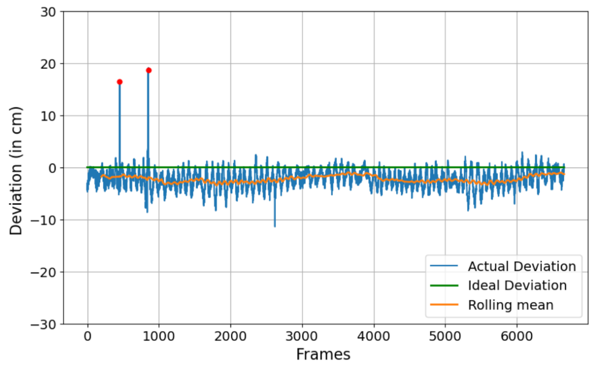


(b)

Row 1: Autonomous run - Anticlockwise direction

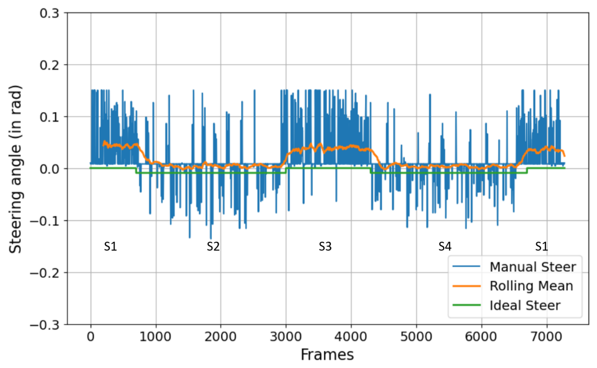


(c)

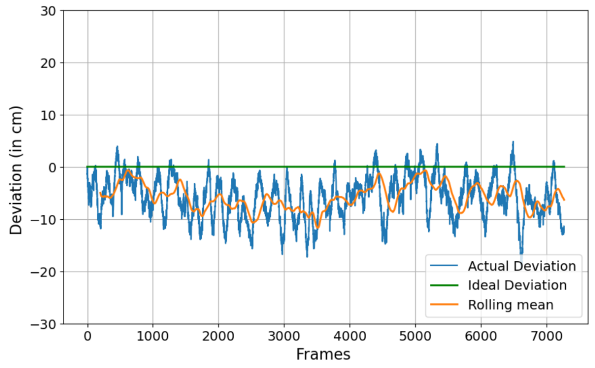


(d)

Row 2: Autonomous run - Clockwise direction



(e)



(f)

Row 3: Manual run - Anticlockwise direction

Fig. 19: Left column: Comparison between ideal steer values and derived steer values; Right column: Comparison between ideal deviations and actual deviations

in an average $MAPD$ value of 1.395 pixels.

As shown in Fig. 18 runs were also performed in different scenarios. As can be seen, the problems of sun glare, road refection, shadows, different colored lanes, and weak lane boundaries were also mitigated. In addition to these, though not intended to specifically address different weather conditions, low lighting, and presence of other lane markers, the algorithm seemed to handle them to a certain extent, indicating the robustness of the implemented system.

2) Derived steering angle: For evaluating the obtained derived steer values, two autonomous runs made in clockwise direction and anticlockwise direction of the track were considered. To facilitate comparison with manual driving, a manual run was also considered. These runs were made in a synthetic lane (red-coloured) of known radius, of curvature, and of known length, and having the mid of the S1 segment as the start and the end point. The steer values were recorded in all the three runs. Substituting the known dimensions of the track and of the kart in Eq. 5, the ideal steer values corresponding to the synthetic lane were computed as zero for the straight segments (S1, S3), -0.0092 for the left turns (S2, S4), and for +0.0092 for the right turns (S4,S2). This ideal steer values were plotted against the recorded derived steer values of the three runs as shown in Fig. 19a, Fig. 19c and Fig. 19e. The rolling mean in the plots are approximations of the derived steer values for understanding the pattern of the curves.

First considering the anticlockwise run vs the ideal values, for S1, the ideal steer is zero; whereas, the derived steer values revolve around 0.05. On analysing, there is a considerable misalignment in the kart's front wheels which drags the kart towards the left side. Additionally, the track has a inward slant as it is an athletic running track. Given these two factors combined, the kart kept significantly moving to the left, and in turn, the algorithm kept producing a right steer to maintain the kart in the center of the lane. This is the constant positive steer observed in the plot for the first segment (half of the S1). Then the steer revolves around 'zero' during the S2 segment. Though this is a left turn segment, given the natural drag of the kart and the inward slant, it was already moving left, hence the algorithm produced minimal left steer. The same pattern repeats in the next straight and the left turn segments S3 and S4 followed by the remaining half of the S1 segment.

In the second plot of the clockwise vs the ideal values, the pattern is the opposite. In the initial half of the S1 segment, the natural left drag of the kart was partly compensated by the inward slant of the lane. For the remaining drag, the algorithm kept on producing a slight right steer around 0.01 as can be seen. This is lesser than what was required for the S1 segment of the anticlockwise run. Whereas in the right turn segment S4, the kart was pulling leftwards but the algorithm needed to give right steer to take a right turn in addition to the compensation. Hence a comparatively larger positive steer is seen throughout that segment. After that, again in the following straight and right turn segments S3 and S2 and the remaining half of S1, the pattern repeats.

Additionally, the kart has an independent suspension system due to which vertical vibrations occur, causing in turn slight vertical vibrations and asymmetric rotations of the camera. Due to this, oscillations keep occurring in the derived steer values as can also be seen in Fig. 19a and Fig. 19c. Simultaneous recording of the data during the autonomous runs is also an important cause for these oscillations as it reduces the responsiveness of the control system. The red dots in the Fig. 19a and Fig. 19c indicate the samples corresponding to the False Negatives discussed as in Fig. 16.

The third plot of the manual drive vs the ideal values is similar to the pattern in the first plot, as the manual drive was run in the anticlockwise direction. However, the magnitude of the steer values is comparatively larger and irregular than that of the autonomous run. Steer values of alternating left and right are seen to occur very often, typically indicative of a manual drive.

3) Behaviour of the kart: The same three runs considered for the derived steer evaluation were considered to evaluate the behaviour of the kart during the autonomous test runs. Primarily, the kart ran autonomously throughout the track but the deviation of the kart from the center of the lane needed to be analyzed to check how effective the steer computation technique is in maintaining the kart at the lane center. This was computed using Eq. 9 for the three runs and are plotted against the ideal deviation as in Fig. 19b, Fig. 19d and Fig. 19f. The plots represent the actual deviation in centimeters from the lane center and hence the ideal deviation is always zero in our case. The rolling mean of the actual deviations represent the approximation of the values.

Comparing the mean deviations for the anticlockwise and the clockwise runs in the Fig. 19b and Fig. 19d respectively, the mean deviation for the clockwise run is slightly lesser than that of the anticlockwise. This is because in the anticlockwise run, both the kart wheel misalignment drag and the inward slant deviates the kart continuously away from the center towards the left; whereas in the clockwise run, the misalignment drag and the inward slant partly compensates each other and hence the actual deviation is comparatively less. However, comparing the manual run in Fig. 19f with Fig. 19b and Fig. 19d, the deviations in the manual run is significantly higher compared to both the autonomous runs. These inferences are also evident from the $MAE\_deviation$ values computed using Eq. 11 for the three runs. For the anticlockwise run, it was 3.216 cm, for the clockwise run, it was 2.370 cm; whereas, for the manual run the $MAE\_deviation$ was calculated as 5.944cm. During the autonomous runs, this deviation was not visible; however, during the manual runs, the deviations were evidently visible.

4) Performance of the kart: Multiple test runs were performed to check the repeatability of the autonomous behaviour. In all the runs, the kart demonstrated the same autonomous capability driving at the center of the lane with minimal deviations, indicating the reliability of the system both qualitatively and quantitatively. The kart did not run off-track in any of the scenarios that the algorithm is designed to handle. In cases of extremely low lighting conditions and sharp shadows, failures were noted; however, the algorithm is not intended to handle these. The algorithm comfortaby operated at 30Hz in all the runs, because of the utilized multi-threading, helping to mitigate erroneous steer generation. The kart demonstrated autonomous driving at varying velocities, still exhibiting the same behaviour.

A notable point of improvement in the algorithm that can improve the performance is the manner in which the algorithm handles a missing lane boundary. For a fixed height of the cameras and a given width of the lane, the sum of θ_1 and θ_2 as seen in the Fig. 9 would remain a constant. Utilizing this, even if only either of the boundaries is detected, a steer value can be estimated. Another way of handling missing boundaries is to utilize a Kalman filter to predict estimates. However, due to time constraints, neither could be implemented in the algorithm and rather the same way of handling as in the SDC implementation was used.

5) Relevance to the RQs: Relevant to the first RQ of this study, the processes and the adaptations implemented in the lane detection techniques showed suitability to the small-scale karts, though this required iterative tuning to make it more generalized. This is a classic disadvantage of traditional real-time approaches.

Considering the second RQ, a steer computation technique as in [12] [15] was used at the SDC and during the initial phases of this follow-up study. The mentioned works claim to maintain the kart at the center of the lane and this technique is used in many literature works. However, real-time demonstration of this technique on a kart or a vehicle is not shown yet. Meanwhile, our study showed that this technique handles only the heading angle error and does not address the cross-track or the lateral error; hence, this technique is not suitable to keep the kart at the center of the lane, contrary to what is claimed. This is an important finding previously not discussed in any of the works that use this technique. This also explains the reason why the kart exhibited oscillating behaviour from the lane center during the SDC challenge, as mentioned in the Results section of the preliminary paper. Therefore, another steer approach as in [2] was adapted and used in our study, which made the kart run in the center of the lane. Even when the kart was placed near either of the boundaries as the start position, the technique was capable of steering the kart to the center of the lane, implying that both the heading and the cross-track errors were handled.

Addressing the third RQ, extensive evaluation was performed for each module of the pipeline which helped relate to the exhibited behaviour of the kart. Proper quantitative evaluation is a gap in the related works, as identified during the literature review. Regarding the fourth RQ, combining the input from multiple cameras was very helpful to augment in case of failures. Though it was used less during this study after the adaptation of the new steer computation technique, this can act as a fail-safe method to keep the kart inside the lane as all the three cameras cannot fail at the same time unless there is a hardware failure or physical occlusions. We also reason that because of this module, the kart managed to stay within the lane during the SDC, despite the deviations from the lane center, implying the significance of this module.

VII. Conclusion

The results presented and elaborated in this follow-up study show that the overall implemented pipeline using traditional lane boundary detection-based approaches is capable of autonomous driving using lane center-following in various driving environments of a small-scale kart. For this purpose, a small-scale kart was successfully adapted and used as the test platform, incorporating a CAN-based communication network and other basic functionalities as the SDC kart. To our knowledge, in no other similar works comprising small scale karts, a communication network similar to that in conventional cars is built or used. Suitability of the pure vision based techniques were investigated and the research questions were answered using this kart. Importantly, a steer computation technique based on the detected lane boundaries, as in a previous work, was identified to be not suitable to maintain the karts in the center of the lane, in contrast to what the work claims. The reason behind the lack of suitability was also identified with experimentation. and this also enabled understanding the reason behind the kart's behaviour during the SDC run. Alternative suitable techniques were also identified, adapted and used to maintain the kart running at the center of the lane. An execution speed of 30 FPS was achieved in real-time tests utilizing multi-threading. Proper quantitative methods for evaluation were identified for each module and the outputs were evaluated using the same. The implemented lane detection techniques achieved Precision, Recall and F1-scores of 0.97, 0.99 and 0.97 respectively, and a value of 1.395 pixels for a newly adapted metric, Mean Average Perpendicular Distance (MAPD), denoting high detection capabilities and closeness of the detections to the ground truth. The derived steer values and the performance of the runs suggest that the autonomous runs outperform the manual runs in running the kart autonomously at the lane center, with a Mean Absolute Error (MAE) of the deviation from the lane center to be around 2-3 cm (towards the left), despite issues with the kart mechanics. Incorporating a PID controller and a model to integrate the discrepancies in the kart mechanics can be a good future improvement to further smoothen the behaviour.

Overall, this traditional approach is fast, reliable and highly suitable for simple applications involving lane lines and can work even with limited processing capabilities. However, this technique works only under the discussed operating domain conditions and does not work when lane lines are not present. Nevertheless, if the processing capabilities are increased, use of AI-based techniques can further expand the working abilities of the approach. To aid in real-time driving scenarios, this technique can very well augment VO-based techniques to adapt autonomous driving to a wider range of operating conditions.

References

[1] I. Kostavelis, E. Boukas, L. Nalpantidis, and A. Gasteratos, "Stereo-based visual odometry for autonomous robot navigation," *International Journal of Advanced Robotic Systems*, vol. 13, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:62266727

[2] V. Umamaheswari, S. Amarjyoti, T. Bakshi, and A. Singh, "Steering angle estimation for autonomous vehicle navigation using hough and euclidean transform," in *Proceedings of the IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, 2015, pp. 1–5.

[3] A. rose Harman, "The environmental benefits of self-driving cars," https://greenerideal.com/news/vehicles/driverless-cars-environmental-benefits/, 2024, accessed: 2024-08-22.

[4] "Self-driving vehicles," https://www.government.nl/topics/mobility-public-transport-and-road-safety/self-driving-vehicles, Government of the Netherlands, 2024, accessed: 2024-08-22.

[5] About RDW. RDW. Accessed: 2024-08-22. [Online]. Available: https://www.selfdrivingchallenge.nl/about-rdw

[6] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022.

[7] evGrandPrix. Purdue University. [Online]. Available: https://engineering.purdue.edu/evGrandPrix/autonomous

[8] Formula Student Driverless. SAE International. Accessed: 2024-08-22. [Online]. Available: https://www.fsaeonline.com/

[9] S. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 1,6, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:114862052

[10] E. Joa, Y. Sun, and F. Borrelli, "Monocular camera localization for automated vehicles using image retrieval," 2021, arXiv preprint. [Online]. Available: https://doi.org/10.48550/arXiv.2109.06296

[11] W. Gates, G. Jati, M. Pratama, W. Jatmiko *et al.*, "A modest system of feature-based stereo visual odometry," in *Proceedings of the IEEE 6th International Workshop on Big Data and Information Security (IWBIS)*, 2021, pp. 47–52.

[12] J. Sujatha *et al.*, "Computer vision based novel steering angle calculation for autonomous vehicles," in *Proceedings of the IEEE 2nd International Conference on Robotic Computing (IRC)*, 2018, pp. 143–146.

[13] M. Aladem and S. A. Rawashdeh, "Lightweight visual odometry for autonomous mobile robots," *Sensors*, vol. 18, no. 9, pp. 2837–2851, 2018.

[14] Y. Tanaka, A. Semmyo, Y. Nishida, S. Yasukawa, J. Ahn, and K. Ishii, "Evaluation of underwater vehicle's self-localization based on visual odometry or sensor odometry," in *Proceedings of the IEEE 14th Conference on Industrial and Information Systems (ICIIS)*, 2019, pp. 384–389.

[15] V. S. Dev, V. S. Variyar, and K. Soman, "Steering angle estimation for autonomous vehicle," in *Proceedings of the IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 871–876.

[16] K. Paponpen, K. Sucharitpongpan, N. Termsaithong, and P. Chaipunya, "The implementation of steering angle estimation on miniature raspberry pi-based autonomous car," in *Proceedings of the IEEE 17th Conference on Industrial Electronics and Applications (ICIEA)*, 2022, pp. 1037–1042.

[17] A. Mahersatillah, Z. Zainuddin, and Y. Yusran, "Unstructured road detection and steering assist based on hsv color space segmentation for autonomous car," in *Proceedings of the IEEE 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 2020, pp. 688–693.

[18] I. Karouach and S. Ivanov, "Lane detection and following approach in self-driving miniature vehicles," 2016. [Online]. Available: https://gupea.ub.gu.se/handle/2077/44673

[19] K. D. N. Tu, H. D. Nguyen, and T. H. Tran, "Vision based steering angle estimation for autonomous vehicles," in *Proceedings of the International Conference on Advanced Technologies for Communications (ATC)*, 2020, pp. 187–192.

[20] M. K. Diab, H. H. Ammar, and R. E. Shalaby, "Self-driving car lane-keeping assist using pid and pure pursuit control," in *Proceedings of the international conference on innovation and intelligence for informatics, computing and technologies (3ICT)*, 2020, pp. 1–6.

[21] J. Á. B. Palma, M. N. I. Bonilla, and R. E. Grande, "Lane line detection computer vision system applied to a scale autonomos car: Automodelcar," in *Proceedings of the IEEE 17th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2020, pp. 1–6.

[22] Y. Ding, "Simple Understanding of Kinematic Bicycle Model," https://shuffleai.blog/blog/Simple_Understanding_of_Kinematic_Bicycle_Model.html, 2024, accessed: 2024-08-22.

[23] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proceedings of the IEEE intelligent vehicles symposium (IV)*, 2015, pp. 1094–1099.