Autonomous Systems Project

Mina Wasfy
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
minawassfy1225@gmail.com

Mohamed Elsaeed
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
mohamed.bargout@student.guc.edu.eg

Nada Tamer
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
nadatameer@outlook.com

Pierre Nader
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
pierre.aprahamian@student.guc.edu.eg

Salma Essam
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
salma.abass2212@gmail.com

Shehab Ashraf
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
shehabguc@gmail.com

Sherif Sameh
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
sherif252525@gmail.com

Ziad Al Wareh
Mechatronics Engineering Department
German University in Cairo
Cairo, Egypt
ziad.alwareh@student.guc.edu.eg

***Index Terms*—Autonomous Systems, Intelligent Systems, Control, Navigation, Localization, Vehicle Dynamics**

I. Introduction

Although some of the earliest work done on the development of Autonomous Vehicles (AVs) can be traced back to the late 1980's through the Eureka PROMETHEUS Project [1] and later on in the 2000's through the Defense Advanced Research Projects Agency (DARPA) challenges. However, the pace at which the field of AVs has advanced at has slowed down over the past two decades. Accordingly, as of the time of writing, only a single car manufacturer, Mercedes-Benz, has a certified production-ready vehicle achieving level 3 of autonomous driving, as defined in the J3016 standard, published by the Society of Automotive Engineers (SAE) in 2014 [2]. Conversely, most other production vehicles achieve at most level 2 of autonomous driving, with level 5 being full autonomous driving with no responsibility or attention required from the driver. Nevertheless, for fully autonomous vehicles to become a reality, there exists many challenges that must be first overcome, of both technical and non-technical nature.

A. Autonomous Vehicle Architecture

Due to the highly challenging and multi-disciplinary nature of the development of AVs; it is common to decompose the problem into the development of separate simpler sub-modules, that are integrated together to form a fully functioning autonomous system. One of the most commonly utilized architectures for the development of AVs can be seen in Figure 1, where the system is decomposed into 4 sub-modules, namely [3]:

- **Perception**: Combining sensor data to recognize and locate both static and dynamic obstacles, interpret road markers, road signs, stoplights, etc.
- **Localization**: Combining sensor data to accurately locate and determine the orientation of the AV with respect to its environment regardless of its surroundings.
- **Planning**: Generating a feasible route to the destination, planning maneuvers such as lane changes, and generating a local collision-free trajectory for the AV to follow.
- **Control**: Controlling the various on-board actuators to track the planned trajectory without exceeding any of the vehicle's physical limits or comprising the passengers' comfort or perceived safety levels.

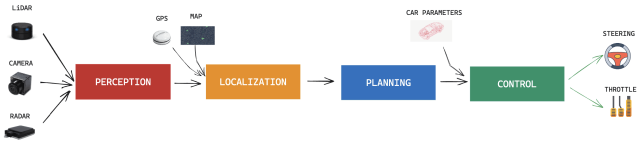

Fig. 1. Autonomous Vehicle Architecture

II. Literature Review

Multiple approaches exist in the literature for tackling the vehicle control problem based on different vehicle models, varying in complexity and accuracy. Moreover, vehicle controllers are often categorized as *lateral* controllers, responsible for eliminating tracking errors, or *longitudinal* controllers, responsible for the vehicle speed control, or *combined* controllers that handle the control of all the vehicle's states. Two

very commonly used lateral controllers are the geometrical controllers Pure-Pursuit [4] and Stanley [5], that rely on the kinematic bicycle vehicle model. In [6], Residual Policy Learning (RPL) was used to amend the base Pure-Pursuit controller in order to improve its tracking performance. Their approach was verified in a simulation environment using the model of a scaled vehicle, which included slip effects. Additionally, in [7], a Pure-Pursuit controller was coupled with a Reinforcement Learning (RL) agent which learns to adapt the lookahead distance in order to maximize corner exit velocity and minimize deviations from the desired path. To verify their approach, the proposed method was tested using scaled vehicles in both simulation and real environments. Conversely, [8] used a linear dynamic bicycle model with linear tires. In their work, a multi-rate Linear Quadratic Regulator (LQR) with an added integral term was used to control the lateral dynamics of the vehicle and verified experimentally utilizing a Hyundai Tucson vehicle. Similarly, the same vehicle model was used in [9], where an artificial potential field approach was used for controlling the vehicle. Accordingly, the derived control forces were a function of the potential field and the deviation of the vehicle from its desired path; such that the equilibrium position coincides with the desired path.

Although many sensors can be used independently for localization such as LiDARs, Cameras, IMUs, Encoders or GPS. However, due to sensor noise and degradation in certain scenarios; pose estimates from different sensors must be fused together in order to perform accurate and robust localization. Accordingly, in [10], wheel and IMU odometry was used to undistort the LiDAR point cloud and provide an initial solution for the LiDAR odometry. Afterwards, the two pose estimates were combined together using an Extended Kalman Fiter (EKF). Similary, in [11], stereo visual odometry was used to provide an initial guess for the LiDAR odometry as well as fusing the heading estimates from the LiDAR and IMU using a KF. Lastly, in order to estimate the vehicle's longitudinal and lateral velocities, an adaptive nonlinear state observer was used in [12]. In their work, the parameter estimates of both the road-tyre friction coefficient and road bank angle where updated online through adaptation laws and their approach showed comparable results to an EKF with a third of the execution time.

Similarly, different approaches for solving the local planning problem exist in the literature for generating collision-free trajectories for the vehicle to follow. An Artificial Potential Field-based (APF) approach was utilized for planning safe and smooth trajectories using Harmonic Potential Fields in [13]. Conversely, in [14], a sampling-based approach in Rapidly-exploring Random Tree Star (RRT*) was used to rectify the pre-planned path to ensure its safety. By using a cost function which penalizes non-smoothness and deviations in curvature from the reference path as well as post processing of the planned path, a path with continuous curvature is generated. Lastly, [15] used a method based on the state lattice technique in order to generate trajectories which are dynamically feasible for the vehicle. Moreover, they showed that their approach could be readily accelerated using a GPU if available.

Considering the vast number of computing devices and running processes on each device that are involved in making an autonomous system, it's not surprising that considerable effort within the literature has gone into facilitating communication and reducing communication delays within these systems. In [16], a platform based on Robot Operating System (ROS) called Intelligent Campus Automobile (iCab) was proposed in order to ease and accelerate the handling of large sensor data, specifically from cameras. Moreover, [17] introduced a cross platform virtual server named "Project Cocktail" to be used for data transmission within the autonomous system. Their proposed platform outperformed ROS in terms of transmission latency and throughput ratio with a similar loss ratio for transferred packets.

III. Methodology

A. Perception

1) Object detection: A 'yolov8n' model was trained to identify different types of obstacles that are frequently seen on roads, The dataset used for training was collected using CoppeliaSim simulation by the vision sensor, the scene used was a track resembling a small part of a city with static and dynamic obstacles like ca. The dataset consists of 1120 images, it was split into 3 parts, 70% of it is used as training data, 20% as validation data, and 10% as testing data. Data preprocessing as well as data augmentation techniques like adding noise or changing hue, were applied to generate a larger set of data. The library utilized for this project is Ultralytics YOLOv$8.2.8$, implemented in Python $3.8.10$ using PyTorch $2.3.0$+cu121 with CUDA acceleration. The pretrained 'yolov8n' model, was retrained over 300 epochs with a learning rate of 0.01 and an image size of $640x640$ pixels, and it was trained on NVIDIA GeForce GTX 1050.

2) Lane detection: To perform lane detection, the raw image from the camera sensor undergoes multiple pre-processing steps. Initially, the image is converted from RGB to a grayscale image, then the brighter features of the image are enhanced by performing a dilation morphological operation. Afterwards, we extract a Region Of Interest (ROI) from the image as defined by a set polygon formed by the chosen pixels and apply a threshold filter to it, where all pixels with a darker intensity than the set thresholds are discarded. After that, the Canny edge detection algorithm is applied to extract edges from the filtered image, from which straight lines are extracted by utilizing the probabilistic Hough Transform. Lastly, we calculate the real coordinates of the vertices of the detected lines through the pixel's corresponding depth values as well as the rigid-body transformations from the camera's frame to the vehicle frame and from the vehicle to the fixed frame.

B. State Estimation

The Kalman Filter (KF) is a linear state estimator requiring a process model as well as a sensor model. This can be obtained

through the state space representation. The states of interest were XY position, heading (ψ), and steering angle (ϕ). The process model as well as the sensor models can be seen in Equations 1 & 2 respectively. Please note that **L** is the length of the vehicle's wheelbase, $\mathbf{V}_{ref}$ is the input reference velocity, and δ is the input steering angle.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \psi_{k+1} \\ \phi_{k+1} \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,0 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \psi_k \\ \phi_k \end{bmatrix} + \begin{bmatrix} -T_s \sin(\psi_k) & 0 \\ T_s \cos(\psi_k) & 0 \\ T_s \tan(\phi_k)/\mathrm{L} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_{\text{ref}} \\ \delta \end{bmatrix} \tag{1}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \psi_k \\ \phi_k \end{bmatrix} \tag{2}$$

KF takes place in five steps:

1) **Predict State:**

$$\hat{X} = AX + BU \tag{3}$$

2) **Predict Error Covariance:**

$$\hat{P} = APA^{\top} + Q \tag{4}$$

3) **Calculate Kalman gain:**

$$K = \hat{P}C^{\top} \left(C\hat{P}C^{\top} + R \right)^{-1} \tag{5}$$

4) **Update State:**

$$X = \hat{X} + K \left(Z - C\hat{X} \right) \tag{6}$$

5) **Update Error Covariance:**

$$P = (I - KC)\,\hat{P} \tag{7}$$

In the KF equations, **P** is the estimate error covariance matrix. **Q** is the process noise covariance, **R** is the sensor covariance, **Z** is the sensor measurements matrix, **C** is the observability matrix, while **K** was the kalman gain.

The state estimator operates by initially subscribing to the ground truth topic. Upon triggering the callback of this topic, position and heading data are extracted and placed into the Z matrix. Gaussian noise is then added to make the sensor non-ideal. Subsequently, the KF callback is invoked. This callback first updates the snapshot of the B matrix and then executes the five KF equations. Upon completion, the final state estimates from the X matrix are extracted and inserted into an odometry-type message, which is then published for use by other nodes. Additionally, the node subscribes to the controller's outputs to receive the input reference velocity as well as the input steering angle, enabling the formulation of the U matrix for the process model. Finally, a ROSbag records all data for post-run analysis, facilitating the validation and tuning of KF parameters.

C. Motion Planning

We simplify the highly complex task of planning obstacle-free, feasible and optimal trajectories, that conform to the road rules, online and in real-time by utilizing a hierarchical architecture for the motion planner. More specifically, the motion planner is broken down into three levels as the following:

1) Global Planner
2) Behavioral Planner
3) Local Planner

1) Global Planner: This is a high-level GPS-like planner which takes as input a desired waypoint that the vehicle should reach as well as the vehicle's current pose. Afterwards, it utilizes the A* graph search technique on an internally defined directional graph of nodes, which represent all the possible interconnections between different roads within the map, in order to find the shortest path from the current position to the goal. Additionally, it handles the limitation imposed on the vehicle by restricting it to forward motion by ensuring that the first waypoints are ahead of the vehicle, relative to its current heading angle.

2) Behavioral Planner: The behavioral planner is responsible for ensuring that the vehicle does not perform any maneuvers which might be feasible but would be breaking the rules of the road such as driving over a crosswalk when a pedestrian is crossing or ignoring stop signs, etc. Although, multiple approaches exist for behavioral planners; however, the most commonly utilized method and the method of choice for this work is the use of Finite State Machines (FSMs). Accordingly, we define a set of discrete states that the vehicle can exist in, namely they are:

- Driving
- Decelerate to Stop
- Remain Stopped

Afterwards, we define our rule base for triggering the transitions between the three states according to multiple factors that involve the vehicle and the environment's current state. Additionally, it simplifies the task of the local planner by processing the global plan and ensuring that it always has a waypoint to target, which falls within a certain defined range in terms of its proximity to the vehicle, as well as specifying whether the vehicle should come to a stop at that waypoint or not.

3) Local Planner: The local planner is responsible for finding an obstacle-free path from the current pose the to the chosen waypoint's pose, then generating a velocity profile for this path which does not invalidate the vehicle's velocity and acceleration limits. To perform this, we utilize the Hybrid A* path planning algorithm as it considers the non-holonomic constraints of the Ackermann model by sampling states from the model's equations of motion. One of the strengths of Hybrid A* is that it considers two heuristics and chooses the one with the greater value, namely they are:

- Non-holonomic without obstacles heuristic
- Holonomic with obstacles heuristic

Such that the former eliminates paths that approach the goal with a wrong heading, while the latter eliminates heavily blocked paths. Additionally, we maintain a local probabilistic map of the likelihood of each cell being occupied using the log-odds representation and update this map iteratively using the incoming data from the perception stack. Lastly, a velocity profile which conforms to the set longitudinal and lateral accelerations as well as enforcing a maximum velocity is generated for this path using a simplified variant of the method known as the Forward-Backward Solver.

D. Control

The vehicle's control is handled by two separate controllers, a lateral and a longitudinal controller, which ensure that the vehicle follows the desired path and tracks the given velocity profile respectively.

1) Lateral Control: The lateral controller utilized in this work is the Stanley Controller, pioneered by Stanford University's Darpa Grand Challenge team. Unlike traditional methods, which primarily focus on the vehicle's direction, Stanley factors in both the car's heading error and its distance from the intended path. It achieves this by using the front axle as a reference point and continuously calculating the cross-track error, representing the deviation from the desired trajectory. The control law for Stanley, which determines the vehicle's steering angle, is calculated according to the following expression:

$$\delta = \tilde{\psi} + \tan^{-1}\left(\frac{k \cdot e}{k_s + v}\right) \tag{8}$$

such that $\tilde{\psi}$ is the heading error, k is a fixed gain and k_s is a softening constant for very low velocities.

2) Longitudinal Control: The speed of the vehicle is controlled through a discrete PID controller, which uses both the throttle and brake inputs to affect the vehicle's speed. The controller calculates a desired acceleration based on a feedforward signal coming from the trajectory as well as the feedback from the PID control law. Afterwards, based on the sign of the acceleration command, either the brake or throttle are commanded with normalized value ranging from 0 to 1.

IV. Results

A. Perception

The trained model was deployed using ROS for real-time inference with data from a vision sensor mounted on a simulated vehicle in CoppeliaSim. A custom ROS node was created to detect objects in incoming images, annotating them with bounding boxes, centroids, and confidence scores. This node outputs an array containing all annotated objects in each image, including their centroids referenced to the map frame and the height and width of the detected bounding boxes in the 3D map. The average publishing frequency of images from the simulator is around 16 Hz, while the average frequency of bounding box publishing is aroud 13 Hz.

The Fig.2 shows a screenshot of the scene with the detected boxes shown in the image.

Fig. 2. Image of the map with detected classes.

The Fig.3 represents the confusion matrix of our model with the trained 14 classes.

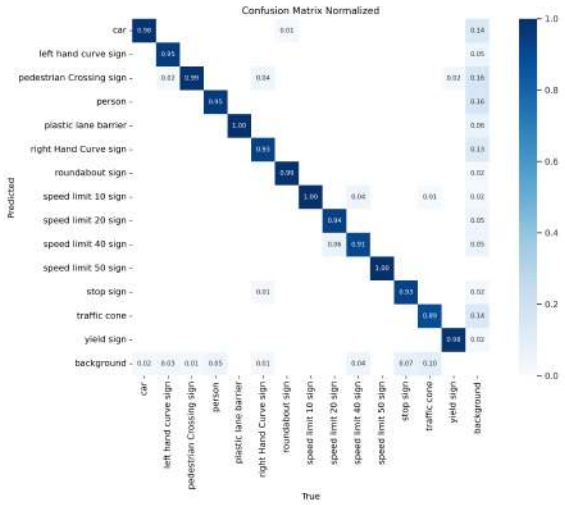

Fig. 3. Normalized Confusion matrix.

B. State Estimation

The performance of the KF is evaluated by adding Gaussian white noise to the ground truth vehicle poses then filtering the noisy readings using the implemented KF. The variance of the noise is set to half the wheelbase and wheel track for the x and y axes respectively. Meanwhile the variance of the heading is set to half the wheel track as well but in degrees. The results for state estimation while tracking an infinity trajectory are shown below in Figures 4 and 5, where its clear that the KF is able to greatly reduce the magnitude of variations in the estimates.

C. Motion Planning

The motion planning is evaluated by firstly testing the global planner's ability to find the shortest path from a start position to a reference position. As shown below in Figure 6, the planner adeptly finds the path connecting the two endpoints.

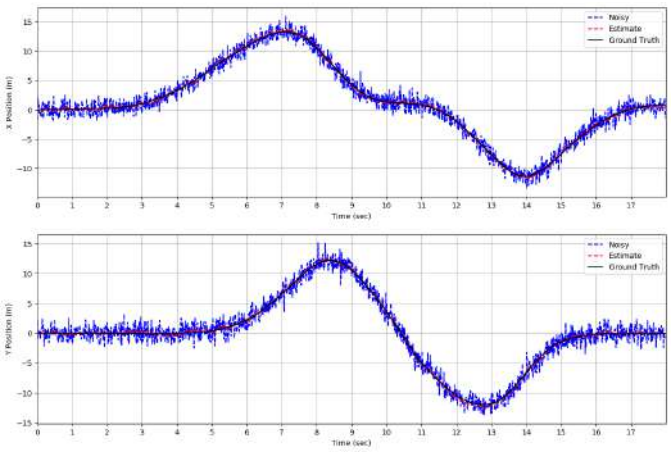

Fig. 4. Position Estimates vs Time

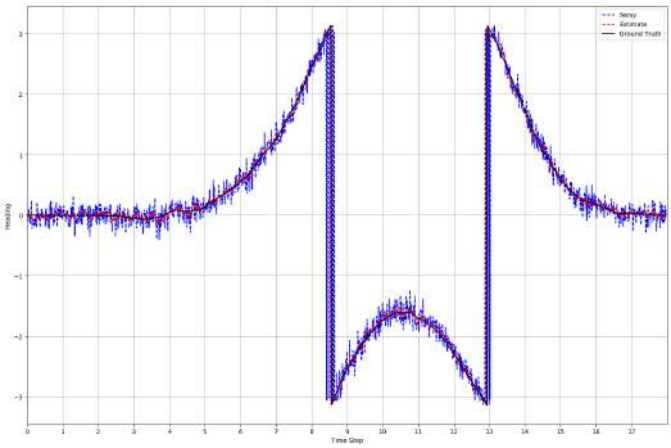

Fig. 5. Heading Estimate vs Time

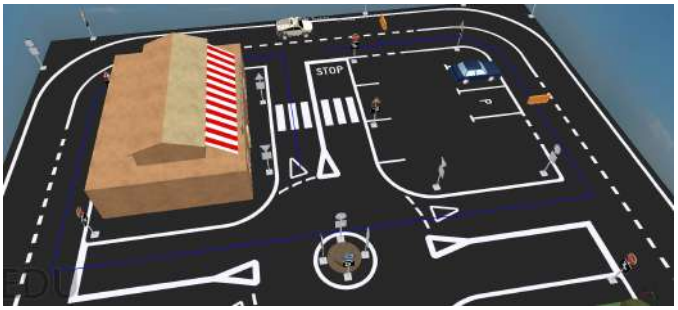

Fig. 6. Global Planner - Planned Path

Afterwards, the local planner was tested by placing a static obstacle in the path to the goal and relying on the planner utilizing the information from the perception stack to navigate around that obstacle. As shown in Figure 7, the planner is able to find a smooth path that avoids the obstacle ahead and does not invalidate the vehicle's non-holonomic constraints.

Fig. 7. Local Planner - Planned Path

D. Control

The controllers were evaluated through tracking of a predefined infinity trajectory so that tracking accuracy can accurately be measured. The results of the tracked path, in the presence of sensor noise, as well as the vehicle's velocity are both shown below in Figures 8 and 9 respectively. As shown in the figures, the vehicle tracks the reference path almost perfectly with minor errors, while maintaining a smooth velocity profile which respects the vehicle's physical limits.

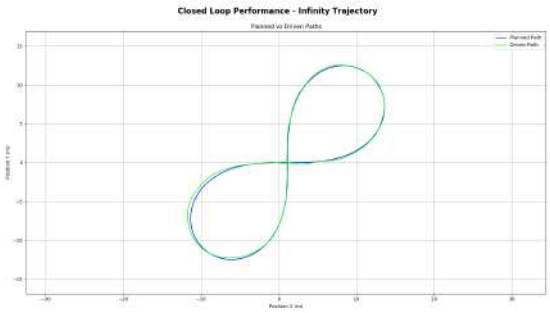

Fig. 8. Planned vs Driven Path

V. Conclusion

To conclude, in this work we have developed a full autonomous system suited to road vehicles, which although limited in its application to static obstacles, is still capable

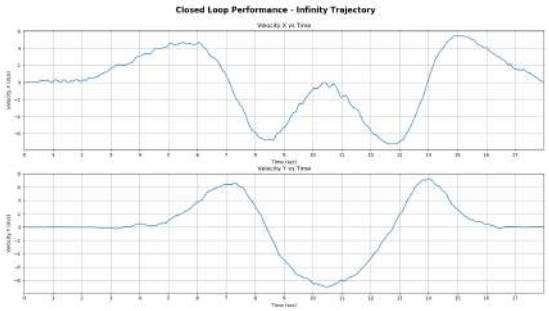

Fig. 9. Vehicle Velocity vs Time

of navigating in a structured and constraining complex city-like scene that creates a realistic driving environment. By utilizing a rigorous testing scheme for each component of the system individually then integrating the different components, we were able to verify the reliability and correct functionality of each component and the system as a whole. Through our testing, we have shown that the overall system is capable of performing full online navigation with no prior knowledge required about its surrounding environment except for the nodes that make up the map in order to substitute for GPS-based navigation.

Finally, in the future we would like to augment the current system with the ability to handle dynamic obstacles such as pedestrians and cars in order to improve the system's reliability and adapt it to more challenging but realistic driving scenarios.

REFERENCES

[1] James Billington. *The Prometheus Project: The Story behind one of Av's greatest developments*. Oct. 2018. URL: https://www.autonomousvehicleinternational.com/features/the-prometheus-project.html.

[2] Bmw. *Autonomous driving – five steps to the self-driving car*. Oct. 2020. URL: https://www.bmw.com/en/automotive-life/autonomous-driving.html.

[3] Jeremy Cohen. *4 pillars vs end to end: How to pick an autonomous vehicle architecture*. Sept. 2023. URL: https://www.thinkautonomous.ai/blog/autonomous-vehicle-architecture/.

[4] Craig Coulter. "Implementation of the Pure Pursuit Path Tracking Algorithm". In: 1992. URL: https://api.semanticscholar.org/CorpusID:62550799.

[5] Gabriel M. Hoffmann et al. "Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing". In: *2007 American Control Conference*. 2007, pp. 2296–2301. DOI: 10.1109/ACC.2007.4282788.

[6] Raphael Trumpp, Denis Hoornaert, and Marco Caccamo. "Residual Policy Learning for Vehicle Control of Autonomous Racing Cars". In: *2023 IEEE Intelligent Vehicles Symposium (IV)* (2023), pp. 1–6. URL: https://api.semanticscholar.org/CorpusID:256846509.

[7] Varundev Sukhil and Madhur Behl. *Adaptive Lookahead Pure-Pursuit for Autonomous Racing*. 2021. arXiv: 2111.08873 [cs.RO].

[8] Young Seop Son et al. "Robust Multirate Control Scheme With Predictive Virtual Lanes for Lane-Keeping System of Autonomous Highway Driving". In: *IEEE Transactions on Vehicular Technology* 64.8 (2015), pp. 3378–3391. DOI: 10.1109/TVT.2014.2356204.

[9] Kirstin L. R. Talvala, Krisada Kritayakirana, and J. Christian Gerdes. "Pushing the limits: From lanekeeping to autonomous racing". In: *Annu. Rev. Control.* 35 (2011), pp. 137–148. URL: https://api.semanticscholar.org/CorpusID:40152710.

[10] Hanzhang Xue, Hao Fu, and Bin Dai. "IMU-Aided High-Frequency Lidar Odometry for Autonomous Driving". In: *Applied Sciences* 9.7 (2019). ISSN: 2076-3417. DOI: 10.3390/app9071506. URL: https://www.mdpi.com/2076-3417/9/7/1506.

[11] Yashar Balazadegan, Siavash Hosseinyalamdary, and Yang Gao. "Visual-LiDAR odometry aided by reduced IMU". In: *ISPRS International Journal of Geo-Information* 5 (Jan. 2016), p. 3. DOI: 10.3390/ijgi5010003.

[12] Lars Imsland et al. "Nonlinear Observer for Lateral Velocity with Friction and Road Bank Adaptation – Validation and Comparison with an Extended Kalman Filter". In: *SAE 2007 Transactions Journal of Passenger Cars - Mechanical Systems* (Jan. 2008). DOI: 10.4271/2007-01-0808.

[13] Robert Daily and David M. Bevly. "Harmonic potential field path planning for high speed vehicles". In: *2008 American Control Conference*. 2008, pp. 4609–4614. DOI: 10.1109/ACC.2008.4587222.

[14] Xiaodong Lan and Stefano Di Cairano. "Continuous curvature path planning for semi-autonomous vehicle maneuvers using RRT". In: *2015 European Control Conference (ECC)*. 2015, pp. 2360–2365. DOI: 10.1109/ECC.2015.7330891.

[15] Matthew McNaughton et al. "Motion planning for autonomous driving with a conformal spatiotemporal lattice". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 4889–4895. DOI: 10.1109/ICRA.2011.5980223.

[16] David Martín Gómez et al. "ROS-based Architecture for Autonomous Intelligent Campus Automobile (iCab)". In: Jan. 2016, pp. 257–272.

[17] Wenhao Zong et al. "Architecture Design and Implementation of an Autonomous Vehicle". In: *IEEE Access* 6 (2018), pp. 21956–21970. DOI: 10.1109/ACCESS.2018.2828260.