Cone detection system with Camera and LiDAR in Formula Student Driverless Car

Adam Paleczny
AGH University of Krakow
Cracow, Poland
adam.paleczny@racing.agh.edu.pl

Rafał Górecki
AGH University of Krakow
Cracow, Poland
rafal.gorecki@racing.agh.edu.pl

Kamil Baradziej
AGH University of Krakow
Cracow, Poland
kamil.baradziej@racing.agh.edu.pl

Piotr Kohut
AGH University of Krakow
Cracow, Poland
pko@agh.edu.pl

2024 15th Annual Undergraduate Research Conference on Applied Computing (URC) | 979-8-3315-2734-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/URC62276.2024.10604539

Abstract - Formula Student is a European engineering competition whose primary goal is building a full racing car. In 2017, Formula Student Germany introduced a new class – Driverless. This category challenges teams to integrate autonomous driving technologies into their vehicles. This paper explores the development of a high-precision cone detection system for a Formula Student Driverless car by AGH Racing, which is crucial for navigating the cone-defined track limits. The entire system was implemented using the Robot Operating System (ROS) framework with Python and C++ programming languages. The developed system is based on the ZED 2i camera and the LiDAR Ouster OS1 128. This approach harnesses the strengths of both technologies: the camera's rapid and accurate image-based detection and the LiDAR's superior precision in generating a 3D point cloud of the environment.

Index Terms—Formula Student, computer vision, LiDAR, Point Cloud Segmentation, YOLO, real-time object detection

I. Introduction

The Formula Student Driverless competition stands at the forefront of advancing automotive engineering and autonomous vehicle technologies, with a specific focus on speed and system reliability. Central of the autonomous capabilities of these vehicles is the cone detection system, which relies heavily on sensor technologies such as cameras and LiDAR's [1][2]. While cameras and LiDAR sensors each have their unique advantages and limitations — often influenced by weather conditions and range detection — their integration promises a significant improvement in the efficiency and accuracy of cone detection systems [3]

Cameras, with their ability to differentiate colors, play a crucial role in identifying the direction of the Formula Student track by recognizing the color-coded cones. On the other hand, LiDAR sensors offer precise localization of these cones within a 3D point cloud, contributing to a more accurate navigation system. However, the true potential of these technologies lies in their fusion, where the complementary capabilities of LiDAR and RGB cameras can be harnessed to achieve unparalleled accuracy and reliability in cone detection under diverse environmental conditions.

This research paper presents the implementation of a synergistic cone detection system utilizing both LiDAR and camera sensors orchestrated through the Robot Operating System (ROS). By exploring the integration of these technologies it aims to enhance the performance of autonomous vehicles in the Formula Student Driverless competition and contribute to the broader field of autonomous navigation systems. The paper delineates the comparative advantages of using both sensors, demonstrating the improved efficiency and accuracy achieved through their combination. This research is separated into chapters about the camera system, LiDAR system, and a summary

II. OVERVIEW

A. Sensors

The system's architecture incorporates two primary sensors: the ZED 2i camera and the Ouster OS1-128 LiDAR, each chosen for their distinctive capabilities and performance characteristics. The Ouster OS1-128 LiDAR is equipped with a high-resolution array of 128 channels, providing an expansive vertical field of view of 45 degrees. Its horizontal field of view is dynamically adjustable through software controls, a feature that, in conjunction with the sensor's rotary mechanism, enables comprehensive 360-degree environmental scanning. This capacity for all-around detection is critical for accurately and reliably acquiring point cloud data in autonomous vehicle applications.

Conversely, the ZED 2i camera is a high-definition 1080p imaging device capable of capturing up to 30 frames per second. It boasts a 120-degree wide-angle field of view, ensuring broad scene capture essential for effective spatial awareness and obstacle detection. Furthermore, proprietary algorithms developed by the manufacturer are utilized to generate a point cloud of three-dimensional points from the captured images.

The entire system operates on the Nvidia Jetson Orin, a computing platform specifically designed for robotics

applications featuring Artificial Intelligence (AI) and Robot Operating System (ROS) optimizations. Owing to its powerful GPU and CUDA-based computation capabilities, processing tasks, especially those requiring intensive calculations, are executed significantly faster.

B. ROS enviroment

Data acquisition from sensors is facilitated through the Robot Operating System (ROS), leveraging official repositories for the ZED camera and Ouster LiDAR to ensure seamless integration and data transformation into ROS-compatible formats. Specifically, the system subscribes to ROS topics to receive image and point cloud data directly from the sensors. The ZED camera is configured to publish RGB images at a frequency of 30 Hz and depth information at 10 Hz. In contrast, the LiDAR sensor operates at a frequency of 20 Hz, ensuring timely and efficient capture of point cloud data.

The architecture of ROS is inherently asynchronous, allowing for the independent and concurrent processing of data streams. This characteristic is particularly advantageous for system, enabling the simultaneous handling of high-throughput data from both camera and LiDAR sensors without significant delays or bottlenecks. The asynchronous nature of ROS underpins the system's efficiency, facilitating real-time processing and analysis of sensor data, which is critical for the dynamic environment encountered in Formula Student Driverless competitions.

III. CAMERA SYSTEM

A. YOLO neural network

1) Models training: You Only Look Once (YOLO) neural network was introduced in 2015 for object detection from photos. It is widely used for real-time object detection [4] from camera sensors. YOLO is a CNN based architecture [5]. Network can make a classification of the objects from pictures and returning bounding boxes - positions (x, y) of the left up left and bottom right points on the picture. An important matter while training own model is dataset. The FSOCO is an open source dataset of cones used in Formula Student [6]. There are about 11 thousand images with bounding boxes. Each BB is classified as one of the following class: ’blue cone’, ’yellow cone’, ’orange cone’, ’large orange cone’, ’unknown cone’ and ’background FK’. Sands for Formula Student Germany rules for Driverless Vehicles [7]. After training several models for YOLOv5 and YOLOv7 [8] were created.

B. Implementation

The integration and deployment of models within the system were achieved utilizing ROS Python and PyTorch frameworks. The process begins with acquiring images via the ZED ROS wrapper, where images are received as ROS topic messages. These images are then preprocessed using the OpenCV library, facilitating their compatibility with the subsequent YOLO model processing. The output from the YOLO model, comprising an array of bounding boxes identifying detected objects within the images, is subsequently published using a custom ROS message format.

Given the computational demands of YOLO processing, particularly noted on the Nvidia Orin platform—the primary computational unit of the presented system — it is observed that processing times were notably extended for each measurement when compared to those on a standard desktop computer. To address this challenge and optimize performance, Nvidia integrated TensorRT into the pipeline. Nvidia TensorRT, a comprehensive SDK tailored for high-performance deep learning inference, includes both an inference optimizer and runtime. This integration is pivotal in minimizing latency and maximizing throughput, significantly enhancing the efficiency of deep learning inference tasks within the presented system.
The example photo of YOLOv5 detection on AGH Racing car is presented on Figure 1:

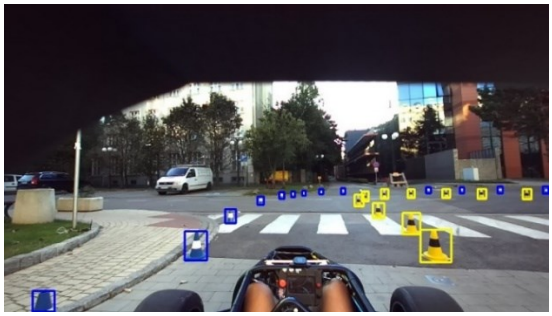

Fig. 1. YOLOv5 working on autonomous car on AGH University of Krakow

C. Depth Camera

ZED 2i is a stereovision camera that uses two cameras and triangulation to create a three-dimensional model of the scene it observes. It estimates depth by calculating the differences between corresponding locations in images taken by two lenses spaced apart by a baseline [9], [22]. Thanks to research paper from Machines Journal [10], it is known that the used camera’s depth is accurate to about 8 meter in Z dimension and has about 0.4-meter accuracy. In the presented system, the Robot Operating System (ROS) facilitates the synchronization of RGB and depth images through the use of the Approximate Time Synchronizer. This mechanism is crucial for aligning the data received from both streams, ensuring that the bounding boxes identified by the YOLO algorithm can be accurately compared against depth images to ascertain the positions of the cones [11]. To estimate the position of a cone, the focal point, defined as the center of the bounding box, is calculated. Subsequently, a ROS message is published, encapsulating an array of cones, each annotated with its class and center position. This methodology enables the precise localization of cones, leveraging the depth information to enhance the spatial understanding of the scene. Such an approach is instrumental for autonomous navigation systems, where accurate real-time environmental mapping is essential.

IV. LIDAR SYSTEM

A. Ground Removal - Algorithm Pathwork++

The precise identification and elimination of ground points from point cloud data constitute an essential preprocessing

phase in the navigation systems of autonomous vehicles. Patchwork++, an enhanced iteration of the Patchwork algorithm, was developed with three fundamental criteria in mind by its creators: speed, the ability to navigate uneven outdoor terrains, and the assurance of both precision and recall in its performance metrics. Given a 3D point cloud, P, which represents the set of measured points, pi, and a snapshot of the surroundings, it is attempt to classify every, p_i, into the ground, including road, terrain, and sidewalks, or non-ground class, which includes cars, buildings, humans, etc. Then, P can be represented as a union of two distinct sets: a set of ground points, G, and non-ground points, N. In contrast, a ground segmentation method outputs the estimated ground points, $\hat{G}$, while the remaining points are regarded as a set of non-ground points, $\hat{N}$. Therefore, P can be expressed as follows [12]:

$$P = \sum p_i = G \cup N = \hat{G} \cup \hat{N}$$

Than the algorithm is divided into several steps:

1) RNR: Reflected Noise Removal

RNR is employed to eliminate noise generated by reflections in the sensor data. Typically, points are filtered out based on a minimum height threshold. RNR scrutinizes a specified number of the lowest rings, targeting points that fall below predefined height and intensity thresholds for removal. [12]

2) CZM: Concentric Zone Model

CZM segments a 3D point cloud into multiple smaller subsets using polar coordinates and varying bin sizes, which are determined by the designated bin zone. This technique facilitates the differentiation of ground and non-ground planes by simplifying the analysis of each segmented part. [13]

3) R-VPF: Region-wise Vertical Plane Fitting R-VPF method is utilized to estimate the vertical plane, particularly in scenarios where the desired ground level is positioned above vertical urban structures. By excluding the vertical plane from the analysis, the accuracy of ground plane estimation is significantly enhanced. [14]

4) R-GPF: Region-wise Ground Plane Fitting

For each region in the CZM, points within each region that fall below a specified seed height threshold are initially identified as ground seeds. Subsequently, the ground plane is determined using Principal Component Analysis (PCA), a technique sensitive to noise, necessitating iterative updates to the estimated plane. Points classified as non-ground, yet lying within a certain distance threshold from the plane, are progressively incorporated into the seed points to refine the plane estimation. [15]

5) A-GLE: Adaptive Ground-Level Estimation and TGR:

Temporal Ground Refinement In this algorithm, the majority of parameters are static and established during initialization. However, A-GLE and TGR dynamically adjust parameters such as elevation, flatness, and noise removal height, leveraging temporal information from previous estimations. This approach is employed to derive parameters that are more closely related to the current environmental context. [12]

More information about Patchwork++ algorithm are based in the research paper [12].

Utilizing the Robot Operating System (ROS) and the ouster ROS package, it was incorporated the Patchwork++ package (available at GitHub [16]) for the efficient segmentation of point cloud data. This package facilitates the division of point clouds into distinct sets of ground and non-ground points. Upon successful segmentation, a ROS message containing the non-ground points is published. This data stream is then utilized for the detection of cones, enabling the accurate identification of these objects within the point cloud.

B. Euclidean Clustering

Following the elimination of ground points, the remaining point cloud is substantially reduced in size, containing far fewer data points. Given the predetermined size and likely positions of the cones, it becomes necessary to discard points that do not correspond to potential cone locations. The only method to achieve this is by iterating through the data and eliminating points that fall outside the probable cone positions.

In the realm of autonomous vehicles, clustering algorithms play a pivotal role in object detection from 3D point clouds due to the wide array of available methods [22]. Among these, the Euclidean Clustering algorithm stands out as particularly notable. This algorithm operates on the principle of spatial proximity, grouping points based on the Euclidean distance between them [17]. The threshold for clustering is determined by the Euclidean distance - d, defined as the distance between any two distinct points. The calculation of d between two points, P_1 and P_2, each defined by their (x, y, z) coordinates, is given by [18]:

$$d = \|P_1P_2\| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Point Clouds consists of thousands of points, which means that the efficiency of searching the target point is relatively low. To satisfy the real-time operational demands of autonomous vehicles, defining a topological relationship among discrete points is essential. This enables the rapid retrieval of points or clusters from their surroundings. That is why KD - tree - multidimensional structure of the binary-tree - is used in the presented system. The KD-tree algorithm partitions the data space into distinct, non-overlapping subspaces, with each node in the tree splitting its assigned data space into two smaller subspaces. Should the data within a node fall below a predefined threshold, the node ceases to subdivide its data space further [19]. Following the extraction of clusters enabled by the high precision of the sensor, it is possible to calculate the dimensions of each cluster. The next step is to determine whether a given cluster corresponds to a potential cone after finding the specifications for cones outlined in the Formula Student Germany handbook [20]. This method relies on comparing the measured dimensions of each cluster against the known measurements of cones, facilitating the identification of cones within the collected sensor data. It is important to remember that only highly precise LiDAR sensors can detect cone colours based on light intensity. In the presented system it can be done for only a limited number of cones. Therefore, the cones identified through used LiDAR processing are not classified.

Cone extraction from point cloud and results is shown on Figure 2:

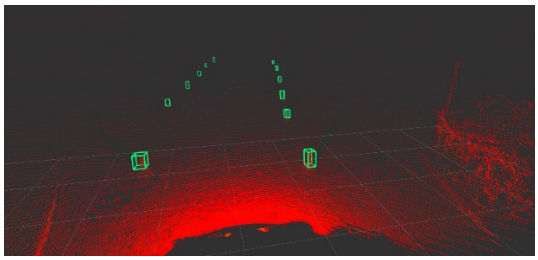

Fig. 2. LiDAR cones extraction

V. ARCHITECTURE

The architecture of the proposed system is comprehensively illustrated in Figure 3, where the diagram delineates the system's modular structure. Sensors are designed by green boxes. In more intense orange are marked ROS packages and with lighter orange elements of calculations in own dv_cone_detector created ROS package. With claret boxes presented are ROS messages consisting array of cones. A detailed exposition of the methodologies employed for both camera and LiDAR-based detection can be found in the subsequent sections of this document. Specifically, Section III - Camera, offers an in-depth analysis of the camera pipeline, and Section IV - LiDAR, similarly delves into the nuances of the LiDAR detection process. Incorporating the Robot Operating System (ROS) infrastructure, the system effectively transforms sensor data into ROS topics using the zed ros and ouster ros packages. For LiDAR data processing, the patchwork package is employed to filter out ground points from the Ouster sensor's ROS topic, streamlining the data for subsequent analysis. It is the implementation of the algorithm from the section IV. It transforms the point cloud from the sensor to two different point clouds of "ground" and "nonground" categories. The primary operations, including the assimilation and analysis of sensor data for cone detection, are consolidated within the dv_ cone_detector package. The system conducts three separate computations, two focusing on cone extraction from camera images. The initial computation retrieves RGB images from the sensor's ROS topic and employs YOLO to identify and publish bounding boxes within the images as ROS messages. Subsequently, the depth camera module cross-references these images with depth information to pinpoint the cones' three-dimensional positions based on the YOLO-determined bounding boxes.

The third component, referred to as Euclidean clustering within the dv_cone_detector package, is dedicated to extracting cones from the "nonground" point cloud. By applying the clustering method detailed in section IV, it accurately determines the cones' positions within the three-dimensional point cloud. As a result, the system produces two sets of potential cone positions. The first set includes cones classified by color, while the second offers more reliable positional data without color information.

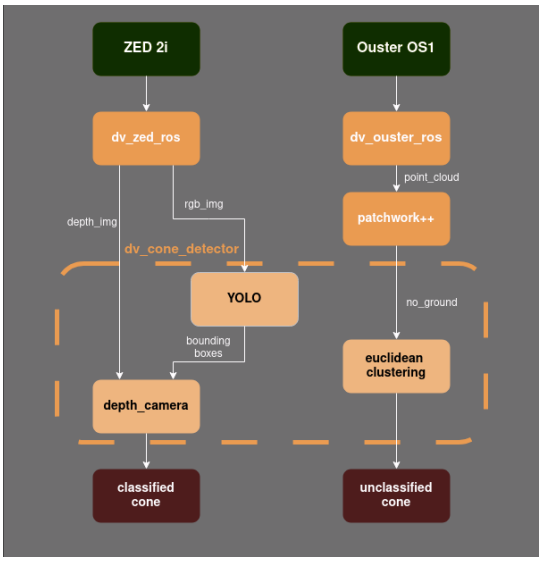

Fig. 3. Architecture diagram of system

VI. TESTING

A. YOLO accuracy

The Confusion Matrix is the most effective method for evaluating the accuracy of AI models for object detection is. In this matrix, rows represent the model's predictions, while columns correspond to actual objects. The matrix entries display the percentage of correct detections, with the diagonal elements indicating accurate predictions. Off-diagonal elements highlight areas where the model most frequently errs, providing insights into its performance and potential areas for improvement. Confusion Matrix for the best models of each Yolo are shown on Figures 4 and 5 . Results from v7 are a bit better than those from v5. Even though prediction of each class is greater or equal 75 %.

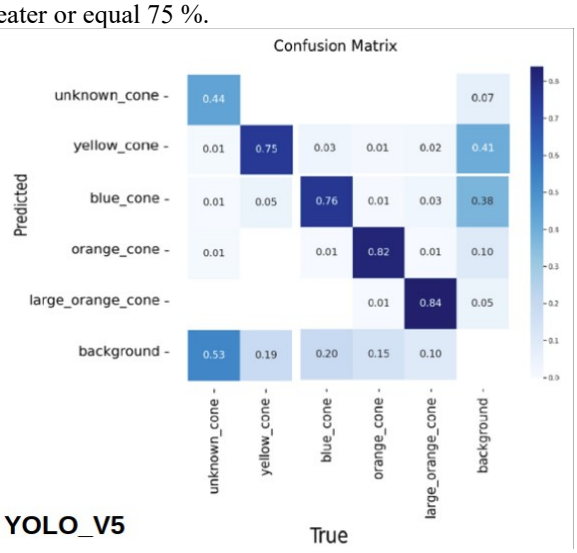

Fig. 4 YOLOv5 Confusion Matrix for self-trained model

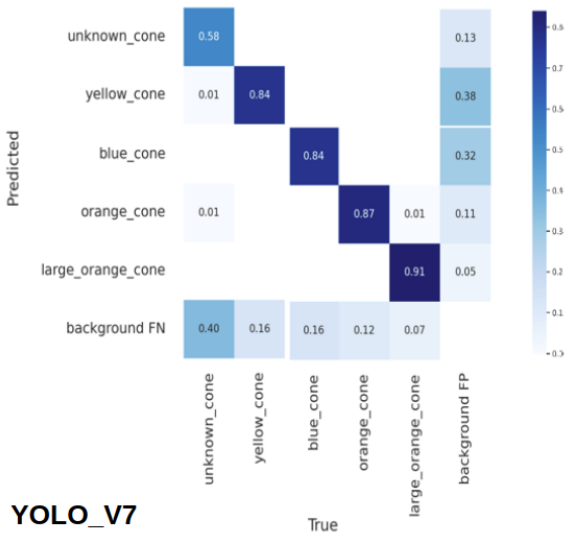

Fig. 5. YOLOv7 Confusion Matrix for self-trained model

B. Computation Time Analysis

To ascertain the efficacy and reliability of the proposed system, comprehensive testing was conducted in both simulated and real-world environments. This dual-mode testing approach was instrumental in evaluating the system's performance across different devices and operating conditions, providing insights into its adaptability and potential applications.

1) Simulation: The primary simulation platform employed was the Formula Student Simulator [21], an open-source tool developed on the Unreal Engine 4 framework and enhanced with the AirSim plugin. This setup facilitated a high-fidelity simulation environment for autonomous vehicle research. Through the integration of ROS bridge, a seamless bidirectional communication was established, allowing for the transfer of image and point cloud data from the simulator to ROS topics, and vice versa for vehicle control commands from ROS to the simulation environment. Performance metrics, particularly the time duration for processing, were meticulously recorded and are summarized in the subsequent Table I. The first table presents data related to the simulation performance on the Lenovo Legion platform with Nvidia RTX 3060, while the second Table II details the results of patchwork++ calculations and Table III shows time needed for non-ground point cloud processing. It is noteworthy that the simulation exhibited accelerated time frames, attributed to the LiDAR's focused detection of only cones and the ground, which simplifies the scene complexity. Additionally, the LiDAR sensor model used in the simulation provided a less precise point cloud compared to that generated by the Ouster sensor, highlighting differences in sensor accuracy and data fidelity.

2) Real-world: Real-world testing was conducted on the AGH University of Krakow campus. The chosen locale, characterized by its diverse mix of buildings and vegetation, introduced a range of variables affecting LiDAR point cloud data quality. These environmental factors were crucial for assessing the system's robustness and its capability to perform in complex scenarios outside of controlled simulation settings. Results of the calculations are shown in Tables I, II, III.

Model	**Calculation time**		
	Fastest time	***Average time***	***Maximum time***
YOLOv5 in simulation	10.92 ms	18.23 ms	23.1 ms
YOLOv5 using TensorRT	7.87 ms	8.14 ms	10.64 ms
YOLOv7 using TensorRT	13.84 ms	14.86 ms	18.67 ms

TABLE I
YOLO DURATION IN SIMULATION AND REAL WORLD

Model	**Calculation time**		
	Fastest time	***Average time***	***Maximum time***
Simulation	1.44 ms	1.51 ms	1.95 ms
Real-world	12.51 ms	12.59 ms	13.37 ms

TABLE II
PATHWORK DURATION IN SIMULATION AND REAL-WORLD

Model	**Calculation time**		
	Fastest time	***Average time***	***Maximum time***
Simulation	1.06 ms	1.23 ms	1.60 ms
Real-world	7.04 ms	7.12 ms	7.29 ms

TABLE III
LIDAR DETECTION DURATION IN SIMULATION AND REAL-WORLD

C. Results

The results indicate that YOLO calculations yield consistent performance in both simulated and real-world environments, affirming the stability of camera-based detection over time, irrespective of external conditions. Conversely, the effectiveness of detection via LiDAR is influenced not only by the sensor's specifications, such as its number of channels, but also by the surrounding environment. The presence of additional structures, like buildings and trees, can significantly extend the time required for cluster extraction, demonstrating the impact of environmental complexity on LiDAR processing efficiency. The computation times achieved for YOLO and LiDAR processing align with the requirements for real-time object detection, successfully identifying cones from each sensor in under 20 milliseconds. This rapid processing time is crucial for effective real-time detection in both everyday and competitive racing environments, where the ability to quickly recognize and react to obstacles can significantly influence the overall safety and performance of the vehicle**.** In our system mostly used is fifth version, because of the faster time for detection. Based on the confusion matrices represented in Figures 4 and 5 for YOLO and the data presented in Table I, it is positioned to select between the more accurate but time-intensive YOLOv7 and the quicker, albeit less reliable, YOLOv5 model for object detection. The choice between these models hinges on the specific environment and the anticipated complexity of the detection tasks. Within the presented system, the YOLOv5 version is predominantly utilized due to its superior detection speed, which is particularly advantageous in

scenarios where rapid processing is essential for real-time responsiveness and decision-making.

VII. CONCLUSION AND FUTURE WORK

This article explores two distinct approaches for cone detection using LiDAR and camera technologies in the context of Formula Student competitions. Each sensor type presents its own set of strengths and weaknesses. The presented system has undergone extensive testing across various environments, both in simulation and real-world scenarios. It can be conclusively stated that while LiDAR-based detection is influenced by the spatial characteristics of the generated point cloud, it tends to yield more precise cone localization compared to camera-based detection. Moreover, custom-trained YOLO models have demonstrated high accuracy in identifying cones within images.

However, these represent just two of many potential methodologies for cone detection. Several strategies exist to improve the accuracy of data analysis further. One such method involves projecting camera images onto LiDAR-generated data, which promises to combine LiDAR's localization precision with the ability to discern cone colors through camera imagery. Another avenue is developing and training Artificial Intelligence models, such as PointNet or PointPillars, to detect cones directly from point cloud data. A critical goal for these new approaches is to maintain or improve upon the processing speed of current algorithms.

The integration of data from multiple detection methods is a pivotal objective for the future. Employing the FastSLAM 2 algorithm will create a comprehensive system capable of autonomous navigation and mapping. This system would not only accurately locate cones but also fuse this information to create detailed and dynamic map of the track environment, significantly enhancing the vehicle's performance in Formula Student competitions.

REFERENCES

[1] Sin-Ye Jhong, Yung-Yao Chen, Chin-Hsien Hasia, Yu-QuanWang, Chin-Feng Lai, ”Desity-Aware and Semantic-Guided Fusion for 3-D Object Detection Using LiDAR-Camera Sensors”, IEEE Sensors Journal, volume 23, 15 September 2023

[2] Zheng Peng, Zhi Xiong, Yao Zhao, Ling Zhang, ”3-D Objects Detection and Tracking Using Solid-State LiDAR and RGB Camera”, July 2023, IEEE Sensors Journa, volume 23

[3] Pan Wei, Lucas Cagle, Tasmia Reza, John Ball, James Gafford, ”LiDAR and Camera Detection Fusion in a Real-Time Industrial Multi-Sensor Collision Avoidance System”, May 2023, MDPI journal Electronics

[4] Lee, J., Hwang, Ki. “YOLO with adaptive frame control for real-time object detection applications”. Multimed Tools Appl 81, 36375–36396 (2022).

[5] Joseph Redmon and Santosh Divvala and Ross Girshick and Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, arXiv, 2015

[6] Vödisch, Niclas and Dodel, David and Schötz, Michael, ”FSOCO: The Formula Student Objects in Context Dataset”, SAE International Journal of Connected and Automated Vehicles, volume 5, 2022

[7] Formula Student Rules for 2024 v1.1, https://www.formulastudent.de/fileadmin/user/upload/all/2024/rules/FSRules2024 v1.1.pdf

[8] T. Reddy Konala, A. Nammi and D. Sree Tella, ”Analysis of Live VideoObject Detection using YOLOv5 and YOLOv7,” 2023 4th International Conference for Emerging Technology (INCET), Belgaum, India, 2023

[9] Saini, V., Kantipudi, M.V.V.P., Meduri, P. (2023). “Enhanced SSD algorithm-based object detection and depth estimation for autonomous vehicle navigation". International Journal of Transport Development and Integration, Vol. 7, No. 4, pp. 341-351.

[10] Tadic, Vladimir Tôth, Attila Vizvari, Zoltan Klincsik, Mihály Sari, Zoltan Sarcevic, Peter Sárosi, József Bíró, István. (2022).”Perspectives of RealSense and ZED Depth Sensors for Robotic Vision Applications.”, Machines. 10. 1-27. 10.3390.

[11] R. Zhang, Y. Yang, W. Wang, L. Zeng, J. Chen and S. McGrath, "An Algorithm for Obstacle Detection based on YOLO and Light Filed Camera," *2018 12th International Conference on Sensing Technology (ICST)*, Limerick, Ireland, 2018, pp. 223-226, doi: 10.1109/ICSensT.2018.8603600.

[12] S. Lee, H. Lim and H. Myung, ”Patchwork++: Fast and Robust Ground Segmentation Solving Partial Under- Segmentation Using 3D Point Cloud,” 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022, pp. 13276-13283, doi: 10.1109

[13] H. Lim, M. Oh and H. Myung, ”Patchwork: Concentric Zone-Based Region-Wise Ground Segmentation With Ground Likelihood Estimation Using a 3D LiDAR Sensor,” in IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 6458-6465, Oct. 2021

[14] S. Wu, L. Ren, B. Du and J. Yuan, ”Robust Ground Segmentation in 3D Point Cloud for Autonomous Driving Vehicles,” 2023 2nd International Conference on Robotics, Artificial Intelligence and Intelligent Control (RAIIC), Mianyang, China, 2023

[15] H. Lim, S. Hwang and H. Myung, ”ERASOR: Egocentric Ratio of Pseudo Occupancy-Based Dynamic Object Removal for Static 3D Point Cloud Map Building,” in IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 2272-2279, April 2021

[16] Patchwork++ code implementation, https://github.com/urlkaist/patchwork-plusplus-ros, last accessed on 9 February 2024

[17] L. Wen, L. He and Z. Gao, ”Research on 3D Point Cloud De- Distortion Algorithm and Its Application on Euclidean Clustering,” in IEEE Access, vol. 7, pp. 86041-86053, 2019

[18] Qu, Jinyan, Shaobin Li, Yanman Li, and Liu Liu. 2023. ”Research on Railway Obstacle Detection Method Based on Developed Euclidean Clustering” Electronics 12, no. 5: 1175.

[19] Z. Xiao and W. Huang, ”Kd-tree based nonuniform simplification of 3D point cloud”, 3rd Int. Conf. Genetic Evol. Comput, Aug. 2009.

[20] Formula Student Germany handbook 2024 for competition, https://www.formulastudent.de/fileadmin/userupload/all/2024/important/docs/FSG24 Competition Handbook v1.0.pdf, last accessed on 9 February 2024.

[21] Formula Student Simulator documentation: fsdriverless. github.io/Formula-Student-Driverless-Simulator/v2.2.0, last entered: 5 February 2024

[22] Wang, Y., Mao, Q., Zhu, H. et al. “Multi-Modal 3D Object Detection in Autonomous Driving”: A Survey. Int J Comput Vis 131, 2122–2152(2023)