A ROS-Based Sensor Fusion Approach to Obstacle Detection and Avoidance for Scaled Vehicles

1st Neelkant Teeluckdharry
Department of Electrical and Computer Engineering
McMaster University
Hamilton, Canada
teeluckn@mcmaster.ca

***Abstract*—In this article, an approach to autonomous navigation is presented using the McMaster Autonomous Electrified Vehicle (MacAEV), a small-scale 1:10 RC vehicle equipped with low-cost sensors. The algorithm solves an optimization problem using integrated data from the onboard depth camera and 2D LiDAR to navigate around detected obstacles. Using ROS 2 middleware, the methodology was implemented and acheived real-time performance.**

***Index Terms*—Autonomous Vehicles, Self-Driving, Data Fusion, Model Predictive Control (MPC), ROS 2 (Robot Operating System), Scaled Vehicles Computational Efficiency, Obstacle Avoidance**

I. Introduction

Autonomous driving has undergone substantial development over the past few decades, driven by researchers and industry partners alike. However, the testing and development of self-driving algorithms on real vehicles is often impeded by the risk posed to human lives. As such, scaled vehicles have emerged as the most suitable alternative for autonomous driving research, permitting researchers to perform rigorous testing of novel algorithms in low-risk settings.

For local path-planning, scaled vehicles leverage algorithms designed for obstacle avoidance, utilizing sensory modules to first detect objects followed by a control strategy to navigate around them. Light Detection and Ranging (LiDAR), an active sensor, is commonly used as a sensory module for these tasks due to the relability of its point cloud data. LiDARs are classified as either 2D or 3D: 3D LiDARs emit multiple laser beams in three dimensional space, while 2D LiDARs produces a single beam at a time and yield a plane scanning. These advanced scanning features of 3D LiDAR scanners can be prohibitively expensive, restricting some researchers from employing scaled vehicles to the field of autonomous driving.

In this paper, the McMaster Autonomous Electrified Vehicle (MacAEV), a scaled vehicle designed at McMaster University (shown in Figure 1), utilizes sensor fusion to detect and avoid obstacles in real-time along a corridor. It is equipped with both a depth camera and a 2D LiDAR, which are interfaced with an embeded computer through Robot Operating System 2 (ROS 2). The data sampled by both sensors is manipulated and fused in C++, and an optimization-based control algorithm, described in Section II, is applied to avoid the detected obstacles.

This project is supported by the National Sciences and Engineering Research Council of Canada

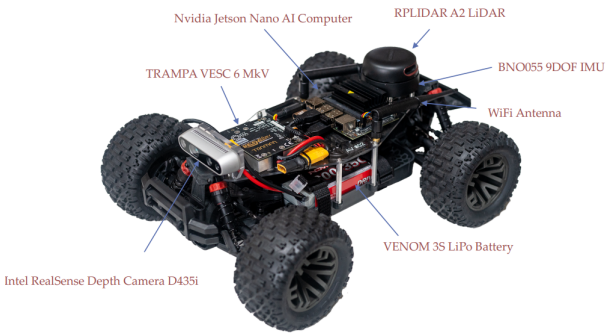

Fig. 1. Shows hardware components of the MacAEV. **Image Credits:** Electrical Systems Integration Project Course.

The paper is structured as follows: Section II examines related work. Section III discusses the algorithms used to detect obstacles and the control strategy used to navigate around them. Section IV examines the hardware components of the MacAEV and interfacing them through ROS 2. Section V discusses the implementation details of the system. Section VI demonstrates the experimental results, showing improved obstacle detection and avoidance and Section VII concludes the paper and presents future directions.

II. Related Work

Obstacle avoidance, a basic task in self-driving, is the backbone of autonomous navigation. Approaches to this task evolved from the Bug Algorithm in favor of path planning algorithms such as A* or Dijkstra on a local cost map [1]. Recently, strategies to this task have now adopted machine learning flavor, giving rise to the optimization-based and model predictive control algorithms. Moreover, a critical aspect of successful obstacle avoidance hinges on accurate detection, for which a myriad of sensors can be employed to continuously scan the surroundings and identify obstacles. The most commonly used sensors for detection are LiDAR and camera, giving rise to LiDAR-based and vision-based approaches respectively [2].

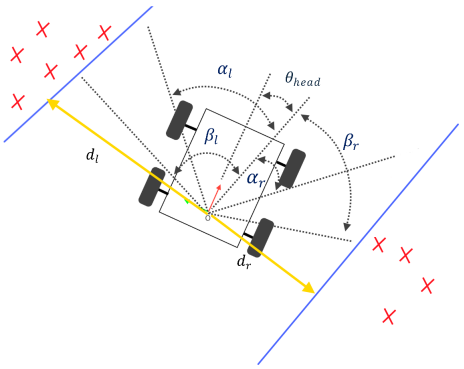

Fig. 2. Vehicle from top-view with labelled parameters used for retrieval of sensory information and control.

Vision-based approaches use either computer vision (CV) or machine learning, specifically Convolutional Neural Network (CNN)-based algorithms such as in [3]. In the context of small-scale autonomous driving, the primary limitation to this approach is the restricted awareness of the surroundings due to the narrow field of vision captured by the camera sampling data. On the other hand, 2D Lidar-based methods, such as the Follow the Gap Method Method described by Sezer et al. in [4], analyze raw returns to detect obstacles. Yet, relying solely on data sampled from a single plane can lead to potential blind spots and missed obstacles. This paper overcomes the limitations of both approaches by proposing a methodology of fusing the data and improves on existing optimization-based control algorithms to navigate around the detected obstacles.

III. Used Algorithms

The algorithm used for autonomous navigation follows the pipeline shown in the Figure.

A. Data Fusion

As stated, a key limitation of 2D LiDARs is their inablity to detect obstacles above and below their scanning plane. One method of addressing this issue involves the incorporation of a depth camera and a fusion of data between both sensors. This is acheived through a frame transformation of points obtained from the depth image to the LiDAR frame, then a comparison of the ranges sampled by both sensors. If the camera measures a point to be closer than its corresponding LiDAR measurement, this means the vehicle is approaching an obstacle below or above the plane of the LiDAR scans. In this case, the corresponding LiDAR scan should be replaced by the camera measurement to give accurate information about the surroundings.

B. Ground Points Filtering

The strategy to fuse LiDAR and camera data is susceptible to errors, as the camera may inadvertently sample ground points due to misalignment, reflective surfaces and other camera intrinsic parameters which create noise. To mitigate this issue, a ground plane is fitted to a selection of the points within a certain threshold using the Random Sample Consensus (RANSAC) algorithm, and the remainder of the points in the image are used for the data fusion. This is described in Algorithm 2.

Algorithm 1: Camera-LiDAR Data Fusion

Data: Depth Image, LiDAR Scans
Result: Fused Ranges
1. Filter out Ground Points from Depth Image
2. Transform each of remaining points to frame of LiDAR
3. If a camera point is closer than corresponding LiDAR point, replace point in fused data

Algorithm 2: Ground Points Filtering

Data: Depth Image
Result: Camera Ranges without Ground Points
1. Randomly select three distinct points in lower half of image
2. Calculate plane coefficients from points
3. Iterate over each pixel, counting number of points within threshold to plane and storing it otherwise
4. Repeat plane generations until enough points have been filtered out or a maximum number of generations has been exceeded

C. Calculating Heading Angle

The fused data is captured and pre-processed as pairs (θ_i, r_i) of beam angles and corresponding range measurements. These scans are processed, assigning beam angles as increments of the LiDAR resolution with respect to the center of a predefined field of vision. This falls within the bound $-\theta_{cen} \leq \theta \leq \theta_{cen}$. Ranges are adjusted to fit the bound $0 \leq r_i \leq r_{max}$, where r_{max} is the maximum LiDAR scan range. This eliminates erroneous data.

The heading angle, θ_{head}, represents the intended direction of travel of the vehicle and acts as a reference to separate scans on the left and right sides of the vehicle in any orientation. It is calculated as a weighted average of beam angles based on the processed data:

$$\theta_{head} = \frac{\sum_i r_i \theta_i}{\sum_i r_i}$$

Two design parameters α_l and β_l, which are angles oriented on the left of θ_{head}, are chosen. The range measurements between these two angles are sampled and used to calculate the left barrier, as described in subsection D below. Similarly, on the right side, α_r and β_r are chosen, and the corresponding range measurements are used to calculate the right barrier.

D. Finding the Barriers

Suppose $w \in \Re^2$ is the vector which parametrizes the line separating the scans from the vehicle. The minimum distance,

d, between the origin of the frame, centered at the LiDAR, and the line is given by:

$$d = \frac{1}{w^T w}$$

The lines on both sides of the vehicle are constrained to be parallel, which better fits the walls of the corridor and smoothen turns around corners. The goal is to find a vector w which maximizes the total distance d_{total}, representing the total distance from the frame to the left barrier d_l and frame to the right barrier d_l.

$$d_{total} = d_l + d_r = \frac{2}{w^T w}$$

This gives rise to a convex model predictive control problem with linear constraints or an inequality-constrained quadratic program.

$$\begin{aligned} \min_{w} \quad & J = \frac{1}{2} w^T w \\ \text{s.t.} \quad & w^T p_i + b - 1 \geq 0 \; \forall i \in \{1, ..., n_l\} \\ & w^T p_j + b + 1 \leq 0 \; \forall j \in \{1, .., n_r\} \\ & -1 + \epsilon \leq b \leq 1 - \epsilon \end{aligned} \tag{1}$$

where n_r and n_l are the number of points sampled between α_r and β_r and α_l and β_l respectively. b is a parameter representing the translation of the lines and $\epsilon > 0$.

E. Steering and Velocity Controller

To drive the motor, the VESC controller is passed an *ackermann_msg* through ROS, which is a time-stamped message containing both the input velocity v and steering angle δ. The MacAEV acheives autonomy through precise control of v and δ within a finite state-machine, where the behaviour of the vehicle changes in response to specific inputs like the detection of an obstacle in-front or behind. The MacAEV's state machine has three states: *normal*, *backup* and *turn*. Developed in this section, are the associated calculations and behavioural descriptions of the vehicle within these states, each of which outputting a corresponding v and δ as *ackermann_msg* to the VESC controller at every time step.

In the *normal* state, the vehicle slows down in the presence of detected obstacles and maintains an appropriate distance from the barriers. Thus, the control objective for steering is to control the distance $d_{lr} = d_l - d_r$ between the left and right barriers. These distances are represented in Figure 2. The non-linear dynamics of these distances are governed by these equations:

$$\dot{d_{lr}} = -v_s \sin \alpha_l - v_s \sin \alpha_r \tag{2}$$

$$\ddot{d_{lr}} = -\frac{v_s^2}{l} (\cos \alpha_l + \cos \alpha_r) \tan \delta \tag{3}$$

Equation (3) arises as a result of the bicycle kinematic model, used to derive the dynamics of the system. Moreover, the tracking error of d_{lr} is minimized using a PD controller, the output of which is used to subsequently calculate the required steering angle δ.

$$\delta_{lr} = \arctan \left(\frac{l(k_p \tilde{d_{lr}} + k_d \dot{d_{lr}})}{v_s^2 (\cos \alpha_r + \cos \alpha_l)} \right) \tag{4}$$

These distances can be expressed in terms of the horizontal components of the vectors found by solving the quadratic program in the previous section. Specifically,

$$\cos \alpha_l = \begin{bmatrix} 0 & -1 \end{bmatrix} \vec{w_l} \tag{5}$$

$$\cos \alpha_r = \begin{bmatrix} 0 & 1 \end{bmatrix} \vec{w_r} \tag{6}$$

Combining equations (4), (5) and (6) produces the closed form expression used in the implementation of the controller to find δ.

$$\delta_{lr} = \arctan \left(\frac{l(k_p \tilde{d_{lr}} + k_d \dot{d_{lr}})}{v_s^2 (w_{ry} - w_{ly})} \right) \tag{7}$$

where w_{ry} and w_{ly} denote the y-component of the right and left wall vectors respectively.

As previously stated, velocity is controlled based on distance to an incoming obstacle. This is given by:

$$v = v_{s_0} \left[1 - e^{-\max(d_{min} - d_{stop}, 0)} \right] \tag{8}$$

where v_{s_0} is the maximum vehicle velocity, d_{min} is the minimum Lidar range reading, and d_{stop} is the vehicle stop distance for an obstacle in front of it. It can analogously be seen as a potential field, where obstacles have a high potential and the vehicle is repelled from them. If the velocity reaches below a defined threshold for a certain amount of time, it indicates that the vehicle has stopped, which triggers a state shift to *backup*.

The *backup* state will cause the vehicle to move with the same velocity scale as in (8), however, in the opposite direction. There are two cases: either the pitch measured by odometry will indicate that the vehicle is oriented away from the closest obstacle or the vehicle will continue attempting to move backwards for a defined amount of time, after which the state will shift to *turn* in both cases.

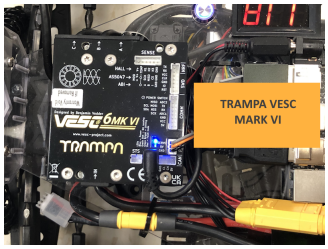

Fig. 3. Shows the TRAMPA VESC MARK VI attached on the vehicle body.

Fig. 4. Shows the SLAMTEK A2M8 attached on the vehicle extremity.

Fig. 5. Shows the Intel RealSense Depth Camera attached to the vehicle extremity.

The *turn* state is also defined with the same velocity scale as in (8). If the vehicle is already oriented away from the closest obstacle, then a state shift will be triggered to return to *normal*. Otherwise, the steering angle will be set to maximum and if the vehicle is still unable orient itself away from obstacles, then it will return to the *backup* state.

IV. System Design and ROS Integration

In this section, the hardware architecture of the MacAEV and ROS interfacing is discussed.

The McMaster AEV car, powered by a VENOM 3S lithium-polymer (LiPo) Battery, is built on an ARMA GRANITE 4x4 vehicle platform and consists of a multitude of sensors and actuators to interact with the environment. These components are connected to an Nvidia Jetson Orin Nano embedded computer, running Ubuntu 20.04 with ROS Foxy middleware.

The system inputs are:

- **SLAMTEK A2M8 RPLIDAR** (Figure 3): Communicates with UART, a sampling rate of 8 kHz, angular resolution of $0.225°$, serial baudrate of 115200 bits per second [6].
- **Intel RealSense Depth Camera D345** (Figure 4): Captures images at 90 fps, resolution of 1280 x 720, uses internal vision processor to correct image captured by left and right imagers [7].

The system outputs are:

- **TRAMPA VESC MKVI** (Figure 5): Motor driver responsible for controlling Brushless DC (BL DC) motors and servos. Built-in wheel odometry, yielding velocity and orientation of vehicle.

The sensory modules and motor driver interface with the embedded computer through ROS-based drivers. As shown in Figure 6, data obtained from the inputs is published through specific topics, namely, LiDAR scans and camera images are published to the **/scan2** and **/camera/depth/image_rect_raw** topics respectively. To make the LiDAR data easier to manipulate for navigation, it is passed through the **experiment** node and republished through the **/scan** topic.

The node responsible for obstacle avoidance is **navigation**. It is subscribed to **/scan**, **/odom**, **/camera/depth/camera_info** and **/camera/depth/image_rect_raw**. Using the described algorithm, it publishes ackermann msgs, meaning both a velocity and steering angle, to the **/drive** topic. The **ackermann_mux** node is subscribed to this topic, which publishes these commands to the **/command/motor/speed** and **/command/servo/position** topic when the vehicle is in autonomous navigation mode. These topics are directly modify motor speed and servo position.

V. Experimental Results

To evaluate the approach, an experiment was carried out in the corridor using boxes of the types shown in Figure 7. For simplicity, they will be referred to as either Type I or II boxes, based on their detectability by the sensory modules.

As seen in Figure 8, the obstacles are placed along the corridor in the order Type I, Type II, Type II, Type I. These obstacles will be referred to as obstacle I, II, III and IV respectively. Each trial begins with the car placed at the start point where autonomous navigation is activated. Then, The car is observed in both the real-world and within the rqt environment, allowing real-time monitoring of both detection and avoidance. The trial ends once the car avoids each of the obstacles in reverse order and returns to the start point.

Ten trials of the experiment were performed without fusing the sensor data. Obstacles which had been displaced by collisions during the initial approach were excluded on return from the final results. The same process was repeated with the car navigating autonomously using fused sensor data.

Using only LiDAR data, obstacle I and IV were never detected while obstacle II and III were sucessfully detected in all of the trials. Using the data fusion pipeline, obstacle I, II, III

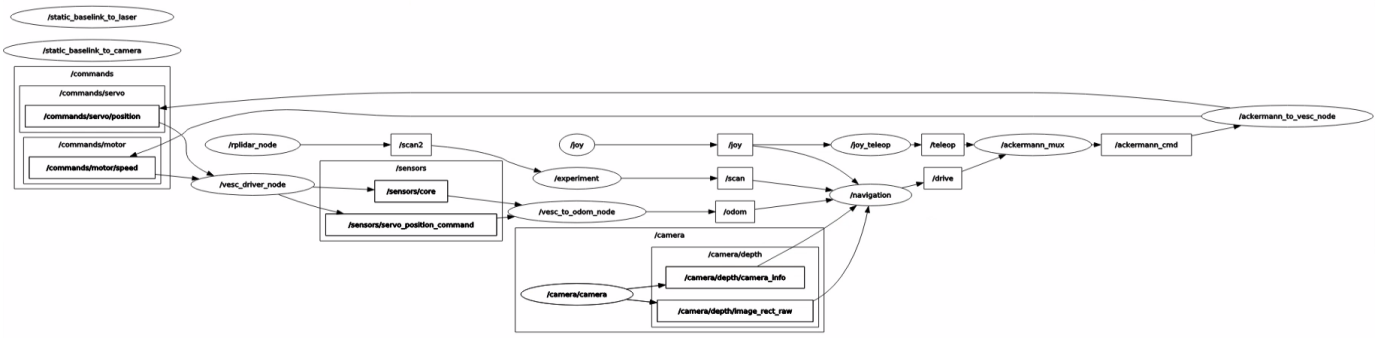

Fig. 6. Shows ROS software architecture of the system, produced by rqt. Nodes are depicted by ovals, while topics are boxed. This architecture is modified from University Pennsylvania's F1Tenth system.

Fig. 7. Obstacles used to evaluate detection and avoidance. a) Type I obstacle that is only detectable by camera. b) Type II obstacle that is detectable by both LiDAR and camera.

Fig. 8. Corridor used for experimental trials.

and IV were detected in 95%, 100%, 100% and 94.1% of trials respectively. Furthermore, using only LiDAR data, the avoidance rates for obstacles I, II, III and IV were 0%, 95%, 100% and 0% respectively. On the other hand, using the data fusion pipeline, the avoidance rates for obstacles I, II, III and IV were 95%, 95%, 100% and 76.5% respectively. This indicates a clear improvement in both detection and avoidance using the data fusion pipeline.

VI. Conclusions and Future Work

In this article, a data fusion approach to obstacle detection was proposed for the MacAEV, combining measurements obtained from a 2D LiDAR and a stereo depth camera. The inherent noise present in the depth camera measurements required additional filtering, which motivated the described RANSAC ground plane approximation algorithm. Using the fused and filtered measurements, an optimization problem was solved, yielding the vectors parametrizing the lines separating obstacles from the vehicle on both sides. These vectors were used in a finite state machine controller to facilitate autonomous navigation. The approach, implemented in C++, worked in real-time as demonstrated by the results.

Overall, this strategy to local planning represents a proposed alternative to laser depth scanning, promising substantial cost savings for both researchers and partners when prototyping self-driving algorithms. Future work will focus on improving both the software and hardware architecture of the system, testing performance in more complex scenarios and feasibility studies of the approach.

Acknowledgment

The author of this paper extends his gratitude to the National Research Council of Canada for supporting this project through funding, as well as Dr. Shahin Sirouspour and other members of the Telerobotics, Haptics and Computational Vision laboratory for their support.

References

[1] Susnea, I., Minzu, V., Vasiliu, G.: Simple, real-time obstacle avoidance algorithm for mobile robots. In: International Conference on Computational intelligence, man-machine systems and cybernetics. World Scientific, Engineering Academy and Society. pp. 24–29. WESEAS'09 (2009)

[2] D. Li, P. Auerbach, and O. Okhrin, "Towards Autonomous Driving with Small-Scale Cars: A Survey of Recent Development.," CoRR, vol. abs/2404.06229, 2024.

[3] S. Badrloo, M. Varshosaz, S. Pirasteh, and J. Li, "Image-Based Obstacle Detection Methods for the Safe Navigation of Unmanned Vehicles: A Review.," Remote. Sens., vol. 14, no. 15, p. 3824, 2022.

[4] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: 'Follow the Gap Method'.," Robotics Auton. Syst., vol. 60, no. 9, pp. 1123–1134, 2012.

[5] “Jetson AGX Orin developer KIT User Guide,” NVIDIA Developer, https://developer.nvidia.com/embedded/learn/jetson-agx-orin-devkit-user-guide/index.html

[6] RPLIDAR A2 Low Cost 360 Degree Laser Range Scanner, https://www.slamtec.ai/wp-content/uploads/2023/11/LD310 _SLAMTEC _rplidar _datasheet _A2M12_v1.0 _en.pdf

[7] A. Grunnet-Jepson et al., “Intel-Realsense D400 Series Datasheet,” Intel RealSense Technology

[8] D. Goldfarb and A. Idnani, “A numerically stable dual method for solving strictly convex quadratic programs,” Sep. 1983, doi: 10.1007/bf02591962.