2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

A Lightweight Approach to Efficient Multimodal 2D Navigation and Mapping: Unified Laser-Scans as an Alternative to 3D Methods

Ocean Noel[1,*], Rafael Cisneros-Limón[1], Kenji Kaneko[1] and Fumio Kanehiro[1]

Abstract— **In this paper, we propose a novel approach for efficient 2D navigation using a multimodal sensor fusion technique. Our method focuses on merging data from multiple sensors, such as LiDARs, cameras, and ultrasonic sensors, into a unified Laser-Scan, which serves as a foundation for faster and more lightweight navigation. By fusing sensor data at the Laser-Scan level, our approach enables the use of basic 2D Simultaneous Localization And Mapping (SLAM) algorithms for mapping tasks, or any others Laser-Scan based features, while still benefiting from the rich information provided by multimodal 3D inputs. This results in a more computationally efficient solution compared to traditional 3D methods that rely on depth points or full multimodal SLAM systems. Our experimental results demonstrate that the proposed approach achieves comparable accuracy in mapping and localization while significantly reducing computational complexity and processing time. This research offers a promising alternative for real-time 2D navigation in resource-constrained autonomous systems, such as drones or any small unmanned vehicles.**

Index Terms— **Sensor fusion, Multimodal sensors, SLAM, Navigation, 2D mapping, Real-time processing.**

I. Introduction

Autonomous navigation in complex and dynamic environments is a critical challenge for various applications, including robotics, self-driving vehicles, and drones. Accurate mapping and efficient localization are essential components of any navigation system, enabling autonomous agents to understand their surroundings and make informed decisions. For addressing these challenges, Simultaneous Localization and Mapping (SLAM) has emerged as a popular solution, particularly with the advent of 3D SLAM techniques that leverage depth information from sensors like 3D LiDARs, stereo cameras, and RGB-D cameras [1]. However, 3D SLAM methods as well as most current navigation features considering 3D obstacles, often require substantial computational resources [2], which can be a limiting factor for resource-constrained autonomous systems. Moreover, processing data from multiple sensors in real-time can be challenging, leading to delays and reduced performance [2]. To address these issues, we propose a lightweight alternative to 3D methods such as those using depth points. Throughout this paper, the term 'Laser-Scan' will refer to the LaserScan message in the Robotic Operating System (ROS) [3][4], a standard message type used to publish 2D LiDAR data. This format typically stores data in a simple list containing the ranges (distances) of detected obstacles or INFINITY value if no obstacle is detected. The position of a range in this list allows to know the direction of the obstacle, based on the Field of View (FoV) of the sensor specified in the Header of the LaserScan message. The LaserScan message is widely used in robotics applications for tasks such as mapping, localization, and obstacle detection, due to its ability to provide a detailed representation of the environment surrounding the robot [5]. Our method focuses on fusing sensors data at the Laser-Scan level, creating a unified Laser-Scan that serves as a foundation for faster and more lightweight 2D navigation.

When navigating in a 2D space, it is still important to consider 3D obstacles, as their shape might not be detected by classical 2D LiDARs and the robot might collide with them. This is particularly true for objects with a shape that changes with the height (i.e a chair or a person), since 2D LIDAR only detects a single layer at a time, whereas a camera can yield 3D information. This allows the system to consider shapes and obstacles outside the 2D LiDAR's plane, addressing the problem that the robot may be taller than the clearance under certain objects. Therefore, we aim to integrate these 3D data into our 2D mapping and navigation system.

Several methods are possible for this integration. We can use usual 3D mapping/navigation algorithms like `RTAB-Map` [6] or `ORB-SLAM3` [7] to represent the environment and project the map [8][2]. However, this can add unnecessary computing time, as Visual SLAM has higher computational requirements than LiDAR SLAM [2]. Also considering 2D navigation, using 3D LiDAR based algorithms like `SC-LeGO-LOAM` [9] may also lead to unnecessary 3D data computing and storage. Other solutions have already been developed for 2D navigation considering the 3D environment [10][11]. Compared to those solutions, that will be detailed in Section II, we propose an easy-to-plug modular package implemented in ROS2 Humble [12][13]. It takes an unrestricted amount of inputs formatted as Laser-Scans, such that any robot can easily plug and play this package into their current mapping, or navigation, system using several different sensors. For example, the result of `SC-LeGO-LOAM` [9] can easily be adapted and used as inputs to our package for lighter high-level computations. Also, the Laser-Scan called "virtual 2D scan" generated from Wulf's system [10][11] can be used with our package and fuse with "other sensor data.

Laser-Scans are the lightest 360-degree representation of the current closest environment around the robot. By converting everything into Laser-Scan prior to any mapping or navigation algorithm, we can avoid the need of heavy 3D approaches. Also, many Laser-Scan-based algorithms already

[1] CNRS-AIST JRL (Joint Robotics Laboratory), IRL, Tsukuba, Japan.
[*] Corresponding author E-mail: ocean.noel.jp@gmail.com

exist in the literature for mapping [14][15][16][17][18], or for navigation purposes [19][20][21][22][23][24][25]. Thus, having a rich Laser-Scan message containing 3D-sourced data can leverage all the existing features using Laser-Scans and lighten the entire navigation stack.

The remainder of this paper is organized as follows:

- Section II provides an overview of the related work.
- Section III details the problem statement; that is, the current limitations.
- Section IV proposes a methodology for fusing sensor data at the Laser-Scan level.
- Section V presents and discusses the experimental results and evaluations.
- Section VI concludes the paper and outlines future work.

II. Related work

A. Sensor Fusion

Sensor fusion is a crucial technique in robotics that combines data from multiple sensors to enhance feedback accuracy and reliability. By integrating complementary information from diverse sensors, such as LiDARs (Light Detection and Ranging), cameras, and IMUs (Inertial Measurement Unit), sensor fusion mitigates individual sensor limitations and uncertainties. This process often employs probabilistic robotics principles, using Bayesian inference and statistical methods to estimate the system's state. Fusing data reduces noise, handles sensor failures gracefully, and improves confidence in the estimated state, leading to more reliable performance in complex environments. Traditional methods like Kalman filters, particle filters, and Bayesian networks have been extensively used to integrate data from various sensors. For example, Thrun et al. [26] provide a comprehensive overview of probabilistic methods for sensor fusion in robotics.

Recent advancements in deep learning have introduced neural network-based approaches for sensor fusion, offering improved performance in complex environments. For instance, Chen et al. [27] demonstrate the use of Convolutional Neural Networks (CNNs) for fusing LiDAR and camera data to improve object detection and localization. Similarly, Wang et al. [28] employ deep learning to fuse data from multiple sensors for enhanced navigation and obstacle avoidance.

B. 2D Navigation Using 3D Sensors Data

Multimodal mapping involves creating maps using data from various sensors, leveraging each sensor's strengths to build a comprehensive and accurate representation of the environment. A LiDAR provides distance measurements with centimeter-level accuracy, operates effectively in low-light conditions, and offers a wide FoV, making it ideal for precise obstacle detection and mapping. Cameras offer rich visual information, including color and texture, crucial for identifying landmarks and understanding the environment's context. However, cameras may struggle in low-light conditions and can be affected by environmental factors such as fog or rain. Combining these sensors enhances the robot's ability to navigate accurately and safely in diverse conditions. Techniques such as probabilistic mapping, occupancy grids, and feature-based mapping are commonly employed to integrate multimodal data. The work by Wulf et al. [10] presents a method for integrating 3D data into 2D navigation by projecting 3D point clouds onto a 2D plane, creating a colored occupancy grid map. While this approach allows the robot to consider 3D obstacles while navigating in a 2D space, it relies solely on a 3D LiDAR and does not directly allow merging data from different sensors.

Similarly, the ROS2 plugin `Navigation2` allows the generation of 2D costmaps from 3D depth points and Laser-Scan inputs [19]. These costmaps are then used to perform path planning and navigation. However, this method does not efficiently consider useful-only data (basically the inner part of obstacles can be ignored), making it computationally costly when several points are considered.

III. Problem Statement

While current 3D multimodal mapping and navigation systems offer high accuracy and robustness, they often require substantial computational resources and can be challenging to implement in real-time on resource-constrained platforms. The processing of high-dimensional data from multiple sensors can lead to delays and reduced performance, making these methods less suitable for applications where computational efficiency is critical.

Moreover, most existing optimized methods focus on a single type of sensor. For example, LiDAR-based SLAM methods, such as `LOAM` [29], are highly effective for 3D mapping but do not efficiently integrate data from other sensors like cameras. Similarly, visual SLAM methods, such as `ORB-SLAM` [30], focus on using monocular camera images. This specialization limits the flexibility and adaptability of these systems in diverse and dynamic environments and on various robot's systems.

Currently, there is a lack of ROS packages that allow for an accurate and efficient fusion of data from multiple sensor types using a standard shared representation. Existing solutions often require custom integration and optimization, which can be time-consuming and complex. For instance, the work by Cadena et al. [1] highlights the challenges in integrating different sensor modalities and the need for more robust and generalized fusion techniques.

In summary, the primary challenges in the current state of the art for 2D Navigation include: 1. High computational requirements for processing high-dimensional data from multiple sensors. 2. Lack of efficient and accurate methods for fusing data from different sensor types. 3. The need for a standardized and modular approach to sensor fusion that can be easily integrated into existing robotic systems.

IV. Methodology

In this section, we present the methodology for converting and merging sensor data into a unified Laser-Scan format, which is essential for efficient and accurate navigation and mapping. We have divided this section into subsections to

provide a comprehensive overview of our approach. The first subsection, "All Sensors to Laser-Scan," details the process of converting data from various sensors, such as depth cameras, into a common Laser-Scan format. This focuses on the specific method for transforming a point cloud of depth points to Laser-Scan, addressing the challenges posed by moving cameras in the robot's frame. The subsequent subsection, "Laser-Scan Merger," focuses on the merging of Laser-Scan data from multiple sensors. This involves transforming the data to a common reference frame, synchronizing the data using odometry interpolation, and performing fusion to create a unified representation of the environment. Each subsection provides detailed steps and considerations to ensure the accuracy and reliability of the merged data. Our method is ready-to-use and available on the public GitHub repository `multi-laserscan-toolbox-ros2` [12].

A. All Sensors to Laser-Scan

Our approach involves converting data from various sensors into a unified Laser-Scan format. For instance, depth information from cameras can be transformed into a Laser-Scan-like representation. This conversion simplifies the data processing pipeline and reduces the computational load.

1) Laser-Scan format: The desired Laser-Scan is represented in polar coordinates, where each point is defined by an angle a and a range r. Mathematically, a Laser-Scan can be described as a set of ordered pairs (a_i,r_i), where a_i represents the angle of the $i-th$ point relative to a reference direction, and r_i represents the distance from the sensor to the detected obstacle at that angle. The angle a typically ranges from 0 to 2π radians (or 0 to 360°) for a full 360° scan. This representation is particularly useful for navigation and mapping tasks, as it provides a straightforward way to represent the environment in terms of distances to obstacles in various directions.

2) Depth Points to Laser-Scan: The output of depth cameras is usually a point cloud, and we need to convert it to a Laser-Scan. The point cloud is represented by a list of points, that we call depth points all along this paper, defined by their position x, y, and z in the camera's frame. The safest way of conversion for navigation is to keep the closest points to the robot from the point cloud projected on the floor plan, and retain this data in the Laser-Scan.

Some algorithms are available to perform such a conversion [31][32]. However, to our best knowledge, none of these algorithms can consider a moving camera in the robot's frame when filtering the depth points. For example, our robot, described in the following Section V, is equipped with two cameras, each with a 52° vertical FoV. These cameras are mounted on servomotors, allowing for tilt movement, which enables them to sweep up and down to gather more information. This dynamic capability should not negatively impact the navigation quality. However, existing algorithms impose constraints to the camera specified in the initial or fixed configuration of the camera. A major limitation of this approach is that it prevents the specification of desired obstacle filtering constraints in the world frame.

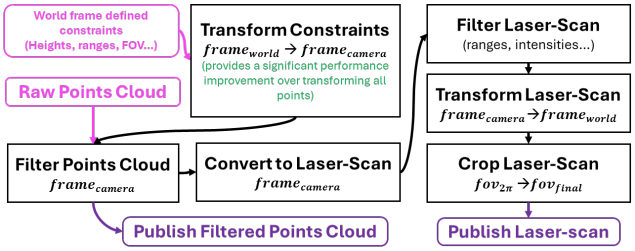

Fig. 1: Point Cloud to Laser-Scan algorithm's Logic.

For instance, consider a scenario where our robot needs to navigate through a doorway with a high overhead clearance. If the camera is tilted downwards to scan the floor, the algorithms may interpret the floor as an obstacle due to the fixed reference configuration. Similarly, if the camera is tilted upwards to scan the ceiling, high objects like door frames might be detected as obstacles, even though the robot could easily pass beneath them. To address this, we developed our own point cloud to Laser-Scan converter that takes into consideration relative motion. Our algorithm allows specifying constraints in the world frame, and using the camera-robot transformation, it automatically extracts the desired filtered point cloud. This allows to consistently filter the wanted depths points even when the camera is tilted up or down. Additionally, our package includes the capability to detect holes and treat them as obstacles. A concise overview of the algorithmic strategy employed by this package is presented in Fig. 1. Although further details cannot be provided in this paper, the package is freely available in our repository [33].

B. Laser-Scan Merger

Once our sensor's data is lightened by being converted into Laser-Scans, they can be used as inputs to our Laser-Scan Merger. An unrestricted number of inputs can be set to the Laser-Scan merger, and each input can be configured individually by specifying the desired FoV, ranges limits (minimum or maximum distances to consider for obstacles), and more. The result is a Laser-Scan containing the synchronized merged information from each sensor. This output can also be further transformed to have a specific FoV or specific ranges. The following subsections will detail the steps involved in this process, including the transformation to a virtual common standard 360° Laser-Scan, the synchronization using odometry interpolation and the global fusion of the data.

1) Transformation to virtual common standard 360° Laser-Scan: Before merging Laser-Scans from multiple sensors, our algorithm standardizes their formats by aligning the resolution (number of points), the FoV, and the reference frame to a common format. Specifically, the common format includes the final desired resolution specified by the user for the output Laser-Scan, a FoV of 360° to encompass data from all possible sensors around the robot, and the output frame specified by the user. The steps involved in this process are illustrated in Fig. 2 and will be detailed in this subsection.

The first step allows to take into consideration Laser-Scans with different resolutions. Indeed, for example, in our case, camera data contains many points, so we decided to convert it into a Laser-Scan with 1440 points, whereas our two LiDARs only have 360 points each. A simple embedded remapping program allows converting a low-resolution Laser-Scan to a higher one and vice-versa. This remapping program also converts the Laser-Scan to a 360° format by adding INFINITY values for out of FoV values, and filter the values according to the constraints fixed by the user (Maximum range, minimum range, specific FoV). The implementation logic for those steps is illustrated in Fig. 2 with, respectively, the names "Resolution conversion", "FoV conversion" and "User preferences Filtering". For the resolution conversion, Eq. (1) is used to determine the new position i_{new} of a polar point in the final Laser-Scan point list of resolution n_{new}, based on its position in the initial Laser-Scan point list i_{init} with n_{init} points.

$$i_{\text{new}} = \frac{n_{\text{new}}}{n_{\text{init}}} \cdot i_{\text{init}} \tag{1}$$

In the case of low-resolution data extended to a higher one, the gaps between the known data are filled with INFINITE range values. These INFINITE values are replaced during the global fusion if another sensor has more data here due to a better resolution or a different origin.

Secondly, merging Laser-Scans from several sensors placed at different origins on the robot requires knowing their relative positions. The algorithm extracts those relative positions from the Laser-Scans directly, as they contain the frame name in which the data are expressed, and transform all the 360° formatted Laser-Scans into virtual common origin Laser-Scans. The transformation involves aligning the data from each sensor to a common reference frame, ensuring that the Laser-Scans are spatially consistent. This means that the positions of obstacles detected by different sensors are correctly mapped relative to each other and to the robot's position, creating a coherent and consistent representation of the surroundings. This step is crucial for accurate fusion of the sensor data. We first transform each Laser-Scan's points from polar coordinates to Cartesian coordinates ((x,y) in 2D). For that, using the definition detailed in Subsection IV.A.1 above, we use Eq. 2.

$$x = r \cdot \cos(a) \qquad y = r \cdot \sin(a) \tag{2}$$

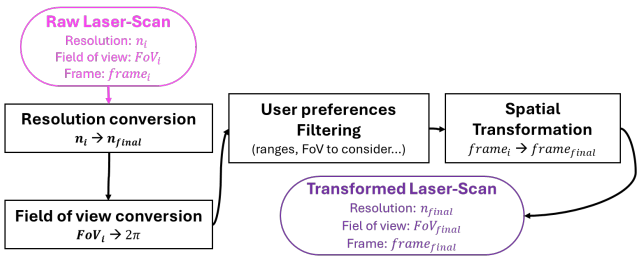

Fig. 2: Algorithm's logic that converts any Laser-Scan to the wanted standard format.

Then, using homogeneous coordinates [34], we transform each Cartesian point to the wanted common virtual frame using the relation Eq. 3, with:

- $A_{|1}$: The homogeneous coordinates of the point A in reference frame R_1.
- $A_{|2}$: The homogeneous coordinates of the point A in reference frame R_2.
- T_{2-1}: The linear homogeneous transformation matrix from R_2 to R_1.
- R_{2-1}: The angular homogeneous transformation matrix from R_2 to R_1.

$$A_{|2} = (T_{2-1}R_{2-1})A_{|1} \tag{3}$$

And then, we transform each Cartesian point in R_2 back to polar coordinates, this time with respect to the new frame, using the relations 4.

$$r^2 = x^2 + y^2 \qquad \tan(\alpha) = \frac{y}{x} \tag{4}$$

This step corresponds to the block named "Spatial Transformation" in Fig. 2.

Finally, the overall algorithm to have compatible Laser-Scans matching the user preferences is as shown in Fig. 2.

2) Laser-Scan Synchronization Using Odometry Interpolation: Format compatibility problems have been solved as described in the previous subsection. However, each sensor might publish its data asynchronously with different rates and with some delay. So the data are not time-compatible and cannot yet be merged as they are. To perform an accurate and consistent fusion, we need to synchronize all the data on a given timestamp. This is a well-known problem when dealing with distributed sensors [35].

In order to deal with that, our algorithm is able to apply a transformation on the received Laser-Scans. If the Laser-Scan message is late, it will apply the correct transformation to it so that even if the data is old, we can extrapolate it to the current timestamp. For that, the algorithm uses the robot's odometry to estimate the motion between the data's timestamp and the current timestamp. However, using the odometry data as a Laser-Scan synchronizer can be problematic. Indeed, as it is also published asynchronously, we might not have the robot's state at the exact timestamp of the sensors' data. Thus, our algorithm contains an embedded odometry interpolator to estimate the robot's state at the time of the sensor data, knowing the state before and after. This interpolator uses the algorithm Slerp [36] for quaternion interpolation for orientation and linear interpolation for position. Let's say we have two odometries $Odom_1$ and $Odom_2$, we define x_i, y_i, q_i, t_i, respectively, the position on the x-axis, the position on the y-axis, the Quaternion and the timestamp of $Odom_i$, which represents the robot's state at time t_i. If the package receives a Laser-Scan at a timestamp t_a with $t_1 < t_a < t_2$, our algorithm will estimate the robot's

state (position, orientation) at t_a with Eq. (5):

$$
\begin{aligned}
u &= \frac{t_a - t_1}{t_2 - t_1} \\
x_a &= x_1(1-u) + x_2 u \\
y_a &= y_1(1-u) + y_2 u \\
q_a &= \text{Slerp}(q_1, q_2, u)
\end{aligned}
\tag{5}
$$

The last available odometry will be used if the wanted state is in the future (i.e., when the sensor's data is available before the odometry at the current timestamp). For state requests older than the oldest available odometry, the oldest data will be used. Our algorithm allows the odometry queue size to be specified. Additionally, a sensor data timeout can be set, which determines the maximum allowable delay between the sensor data and the current time, to avoid using too old data.

It is worth to note that the odometry data are usually not fully accurate enough for integration; indeed, small errors in speed measurements can accumulate over time, leading to drift in the estimated robot position and orientation. This makes the odometry data unsuitable for determining the robot's true current state without additional correction methods. But, this problem doesn't affect our Laser-Scan merger as it is based on odometry differences. The error that is introduced is therefore the error between two consecutive odometry data, and this is usually small enough to have good accuracy when transforming Laser-Scans. The full code logic to synchronise a Laser-Scan on a wanted timestamp is as shown in Fig. 3.

3) Global Fusion: Finally, we perform the global fusion of the aligned and synchronized Laser-Scans to create the wanted unified representation of the 3D environment in a 2D Laser-Scan. This step allows combining the data from all sensors to build a rich Laser-Scan that can be used for navigation or mapping. The fusion simply consists of comparing all Laser-Scans and keep the smallest ranges, which represent the closest obstacles. The overall algorithm merging the Laser-Scans from an unrestricted amount of sensors is as described in Fig. 4.

V. Experiments and Results

Our algorithm is implemented as a ROS2 package, making it easily integrable with any existing robot already using ROS2[4]. In our case, we integrated our ROS2 package with the navigation system of an omnidirectional robot called CALL-M. This robot is designed to collect cardboards and trash bags in a shopping mall, requiring it to navigate in an environment filled with complex-shaped objects and dynamic obstacles.

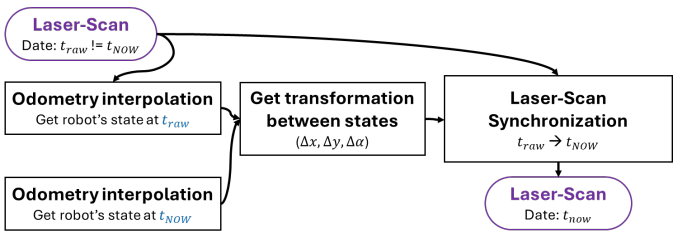

Fig. 3: Algorithm's logic to synchronize a Laser-Scan with current robot's state.

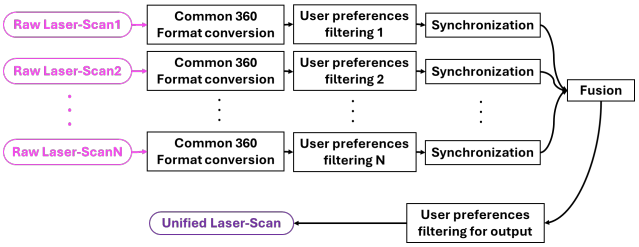

Fig. 4: Algorithm's logic for a consistent Laser-Scans fusion.

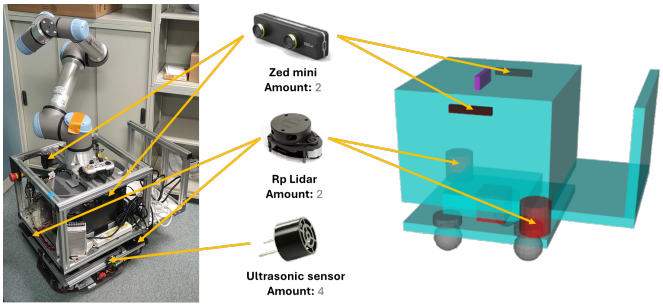

Fig. 5: CALL-M robot's sensors.

To achieve this, CALL-M is equipped with two 2D RpLidars SLAMTEC A1[1], two Zed-Mini[2] cameras, and four Ultrasonic sensors Maxbotic MB1403[3]. Also, the two cameras are mounted on servomotors allowing them to sweep up and down. The Ultrasonic sensors were added later and are not yet implemented in the ROS2 navigation system, so they were not used in the experiments presented here. The sensors were arranged as shown in Fig. 5, with the hardware on the left and the corresponding simulated robot model for the mobile base only (without the Ultrasonic sensors) on the right. Table I outlines how each sensor enhances the robot's environmental awareness and navigation capabilities.

The robot is also equipped with two computers, a Nvidia JETSON Orin NX 16 Gb[4] to manage GPU related packages (especially for the Zed-Minis) and a NUC 13 Pro Kit[5] to run other programs related to control. We use a TriOrb[6] mobile base allowing for omnidirectionnal motion, and UR5e robotic arm[7] for further grasping tasks. The robot is around 85 Kg (with the UR5e), has a rectangle footprint of 0.48 m x 0.74 m, and an height of 0.60 m without the UR5e on top.

The inputs to our algorithm are unrestricted but need to be Laser-Scans. Therefore, the data from the Depth Cameras have been converted to Laser-Scan using our developed

[1]https://www.slamtec.ai/product/slamtec-rplidar-a1/
[2]https://www.stereolabs.com/en-fr/store/products/zed-mini
[3]https://maxbotix.com/products/mb1403
[4]https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/
[5]https://www.asus.com/displays-desktops/nucs/nuc-mini-pcs/asus-nuc-13-pro/techspec/
[6]https://triorb.co.jp/en/
[7]https://www.universal-robots.com/products/ur5-robot/

LiDARs	Provide accurate 2D obstacle detection, robust to luminosity variations. The two LiDARs offer a 360° FoV.
Cameras	Allow 3D detection of obstacles, considering the height of obstacles. However, the two cameras combined cover only a 164° FoV. Also, data is sensitive to luminosity variations, fog, and other environmental factors.
Ultrasonic sensor	Currently not in use, but intended for detection of glasses and mirrors. They detect obstacles accurately within a certain range, though the position accuracy is limited.

TABLE I: Advantages and drawbacks of mounted LiDARs, Cameras and Ultrasonic sensors.

Specification	LiDAR	Camera
Laser-Scan Publish Rate (Hz)	8	14
Laser-Scan Resolution (number of points)	360	1440
Laser-Scan FoV (°)	360	82

TABLE II: LiDARs and Cameras ROS2 specifications.

(a) Simulated environment. (b) Map using LiDARs (c) Map using LiDARs and cameras

Fig. 6: Mapping using LiDARs Only and with Depth Cameras' data integration.

package presented in Section IV. This package appeared to be efficient and could run at more than 300 Hz[8] when processing 52000 depth points from each camera on a CPU i7-7700HQ. Additionally, given the LiDARs' arrangement, we needed to filter their outputs to prevent the robot's body from being detected and also to merge their data. As the LiDARs' outputs are Laser-Scans, we use them directly as inputs to our algorithm. So we can filter and fuse them with the cameras' Laser-Scans into a virtual LiDAR at the center of the robot. Also, our package facilitates this process of individually customizing each source easily through a .yaml ROS2 configuration file.

The performance specifications of each sensor are presented in Table II.

To validate our algorithm efficiency when merging data from two LiDARs and two Cameras, two experiments are presented in this paper. First, we demonstrate how our algorithm enhances 2D mapping by considering 3D obstacles with the cameras' data. Then, we show how our algorithm allows to lighten an existing navigation system while maintaining the navigation quality considering 3D obstacles. Both experiments presented here have been done in a simulated environment in Gazebo. Complications with the hardware of the mobile base prevented us to conduct those final experiments on the real robot. Those experiments also demonstrate that our package seamlessly enables the combination and customization of data from multiple sources.

A. Light mapping with LiDARs and Cameras Using ROS2

Fig. 6 shows that we could correctly fuse cameras' 3D data and 2D LiDARs' data into a rich unified Laser-Scan from which a rich map containing also the 3D obstacles information can be generated through a classic 2D SLAM. In our case `slam-toolbox` package has been used [14]. The left part shows the 3D scene used to perform the mapping. The middle part shows the map before integrating cameras, where we can note that only the footprints of tables and shelves appear on the map. However, considering the robot height and the arm that is mounted on it, we want to avoid the robot to go below these objects; thus, we had to consider 3D camera's data. The right part shows the map resulting from this integration. We can see that this map contains the appropriate obstacles that should be considered according to our robot's height, improving the environment knowledge and proving that we could correctly enhance the classical 2D SLAM using our package. The mapping of the shelves in the top left corner is incomplete because the robot did not navigate between each shelf to perform a thorough mapping during this experiment.

We noted that the generated map when integrating cameras contains thicker obstacles than the one using LiDARs. This effect doesn't appear from our package merging the Laser-Scans, but from our depth to Laser-Scan package converting camera's data to Laser-Scan. This is due to the high resolution of the cameras' Laser-Scan (1440 points like specified in Table II). Due to this high resolution, some inside points like holes between boxes or shelves' support might be considered as the closest points. But this effect doesn't add wrong points and the amount of inner points considered is negligible compared to the true closest point detected. This effect wasn't a problem for accurate mapping and navigation in our case. Anyway, our Depth to Laser-Scan converter package can be adjusted, and the resolution can be reduced to avoid this effect if needed.

B. Light navigation in dynamic Environment with LiDARs and Cameras Using ROS2

We also tested our approach in a dynamic environment where the robot needs to navigate considering 3D data from two cameras and 2D data from two LiDARs. To measure how our strategy allows to lighten the navigation, we used `Navigation2` ROS2 plugin (`NAV2`) [19]. This plugin contains several ready-to-use 2D path planning and controllers. And NAV2 can take as inputs Laser-Scans and point clouds, so we wanted to compare the performance when using our LiDARs' Laser-Scans and cameras' depth points directly as inputs to the default system of `Navigation2`, and when using only our rich Laser-Scan that contains all

[8]The measured frequency is based on the processing time for 52000 depth points. During runtime, the package fits to the maximum rate of the cameras' publications which was 14 Hz in our case.

Model	Acer Predator G9-793
Processor	Intel® Core™ i7-7700HQ CPU @ 2.80 GHz × 8
Memory	20 Gb
Graphic Card	NVIDIA GeForce GTX 1060 Mobile 6 Gb
OS	Ubuntu 22.04.4 LTS

TABLE III: Experiment's computer details.

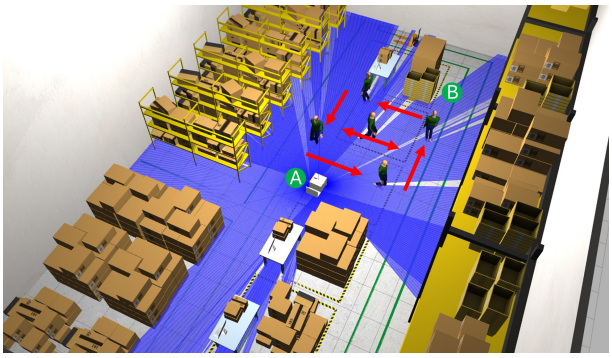

Fig. 7: Simulated world for the Navigation experiment.

the information. We operate the same navigation simulation through the same obstacles and compare the navigation system load and the duration to reach the goal. As shown Fig. 7, the experiment consists of asking the robot to go from point A to point B while navigating through the moving obstacles, represented by simulated humans. The details of the computer used to perform the experiment can be found in Table III.

We operate the experiment several times with different amounts of depth points from the cameras, and for each following method:

- Raw inputs: The two LiDARs and two cameras are directly given as inputs to `NAV2` navigation system.
- Merged Laser-Scan: The two LiDARs and cameras are merged into one Laser-Scan using our algorithm, and only this Laser-Scan is given to the navigation system.

Fig. 8 shows the measured duration to reach the goal, and the different loads (CPU and Memory) for each case.

First, we observe that the navigation duration is unaffected regardless of the method used. This demonstrates that our method maintains good navigation quality. Then, as more depth points are considered, the CPU usage of the default `NAV2` navigation system increases, while our algorithm allows to maintain a constant CPU load. This is because the default system only computes one Laser-Scan when using our method. And the `multi-laserscan-toolbox-ros2` package, processes fixed-size Laser-Scan inputs and has also a constant CPU Load. However, our approach requires an additional conversion of depth points to Laser-Scan using the `depth-filter-scan-converter` package described in Section IV.A.2.

As shown in Fig. 9, we also monitor the overall computer's system loads. A large amount of the measured loads are due to the simulated environment but we can still note a difference when using our method or the default `NAV2` navigation system. Especially with large amounts of depth points, using the default NAV2 system slows down the computer, with a simulation running at 40 FPS rather than the usual 61 FPS, because of the large CPU Load reaching 100%, but our algorithm still allows the computer to run properly considering the same sensors with a maximum load around 97%.

Those results demonstrate the efficiency of merging all sensors into a rich Laser-Scan using our algorithm for a lighter and enhanced mapping or navigation. Our rich Laser-Scan will enhance any higher-level algorithm using it for mapping or navigation.

VI. Conclusion

In this paper, we presented a lightweight approach to efficient 2D navigation using multimodal sensor fusion. Our method is based on converting data from various sensors into a standardized Laser-Scan format and performing fusion to create a rich Laser-Scan for mapping or any Laser-Scan based navigation system. Experiments using ROS2 demonstrated the effectiveness of our approach, showing similar quality navigation and significant reduced computational load compared to traditional 2D navigation methods considering 3D obstacles. Additionally, our approach proved to be fast enough for robustness and efficiency in handling dynamic environments and for enhancing the mapping of a common 2D SLAM with 3D obstacles data.

Our proposed method aims to provide a lightweight and modular ROS2 solution for fusing sensor data at the Laser-Scan level from an unrestricted amount of sensors, enabling efficient and accurate 2D navigation or mapping while considering 3D obstacles. It does not aim to replace existing methods for 2D mapping and navigation but rather seeks to enhance them by reducing their computational load, through the efficient formatting of sensor data into the Laser-Scan format and integrating data from multiple 3D and 2D sensors.

Future work will integrate ultrasonic sensors into the navigation system, requiring further study to convert their imprecise range data into Laser-Scan format. Additionally, future research will explore synergy with the robotic arm to enhance navigation by considering its motion.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] Y.-L. Zhao, Y.-T. Hong, and H.-P. Huang, "Comprehensive Performance Evaluation between Visual SLAM and LiDAR SLAM for Mobile Robots: Theories and Experiments," *Applied Sciences*, vol. 14, no. 9, p. 3945, 2024.

[3] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[4] ROS, "ROS: An open-source robot operating system," https://www.ros.org/.

[5] ——, "Introduction to Working with Laser Scanner Data," http://wiki.ros.org/laser_pipeline/Tutorials/IntroductionToWorkingWithLaserScannerData.

[6] M. Labbé and F. Michaud, "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of field robotics*, vol. 36, no. 2, pp. 416–446, 2019.

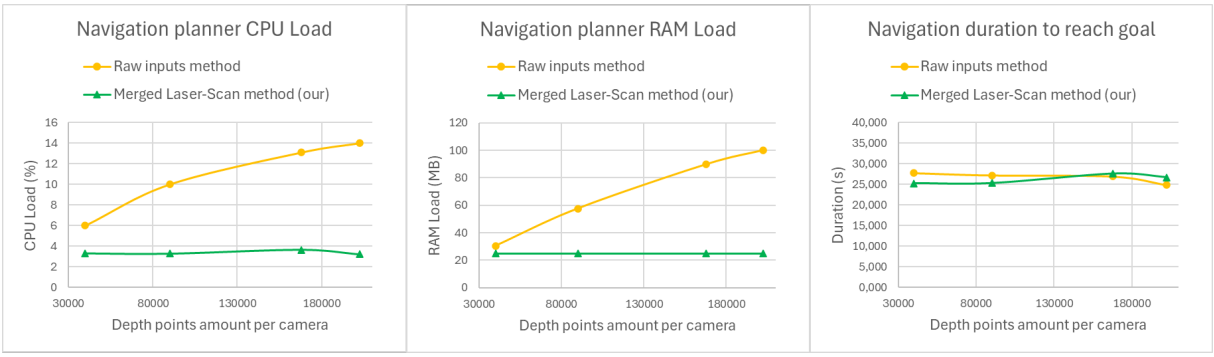

Fig. 8: Duration and Navigation system load for different cameras' amount of points and for each method.

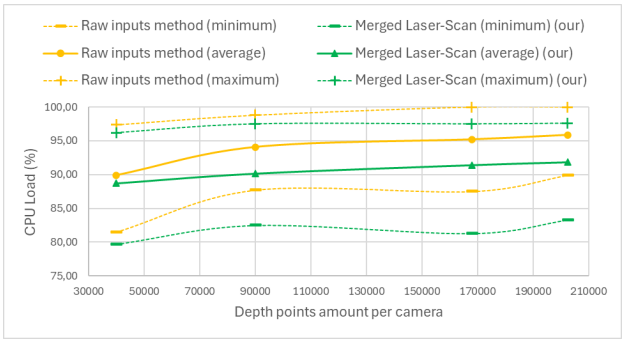

Fig. 9: Overall CPU load during Navigation for different amount of depth points and for each method.

[7] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[8] Y. Nitta, D. Bogale, Y. Kuba, and Z. Tian, "Evaluating SLAM 2d and 3d mappings of indoor structures," in *ISARC. Proceedings of the international symposium on automation and robotics in construction*, vol. 37. IAARC Publications, 2020, pp. 821–828.

[9] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[10] O. Wulf, C. Brenneke, and B. Wagner, "Colored 2D maps for robot navigation with 3D sensor data," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2991–2996.

[11] O. Wulf, K. O. Arras, H. I. Christensen, and B. Wagner, "2D mapping of cluttered indoor environments by means of 3D perception," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 4. IEEE, 2004, pp. 4204–4209.

[12] O. Noel, "multi-laserscan-toolbox-ros2," https://github.com/Noceo200/multi-laserscan-toolbox-ros2.

[13] ROS, "ROS 2 Humble Documentation," https://docs.ros.org/en/humble/index.html.

[14] S. Macenski and I. Jambrecic, "SLAM Toolbox: SLAM for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, 2021, http://wiki.ros.org/slam_toolbox.

[15] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007, http://docs.ros.org/en/hydro/api/gmapping/html/index.html.

[16] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in *2011 IEEE international symposium on safety, security, and rescue robotics*. IEEE, 2011, pp. 155–160, http://wiki.ros.org/hector_slam.

[17] W. Hess, D. Kohler, H. Rapp, F. Andert, and W. Burgard, "Real-Time Loop Closure in 2D LIDAR SLAM," https://google-cartographer-ros.readthedocs.io.

[18] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C)*, vol. 2. IEEE, 1999, pp. 1322–1328, http://wiki.ros.org/amcl.

[19] S. Macenski, F. Martín, R. White, and J. Ginés Clavero, "The Marathon 2: A Navigation System," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, https://docs.nav2.org/. [Online]. Available: https://github.com/ros-planning/navigation2

[20] O. Noel, "ROS2 Vector Field Controller," https://github.com/Noceo200/ros2-vector-field-controller.

[21] R. N. S. Maintainers, "Costmap 2D Documentation," http://wiki.ros.org/costmap_2d.

[22] R. P. Maintainers, "Voxel Grid Documentation," http://wiki.ros.org/voxel_grid.

[23] R. N. S. Maintainers, "Move Base Documentation," http://wiki.ros.org/move_base.

[24] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997, http://wiki.ros.org/dwa_local_planner.

[25] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017, http://wiki.ros.org/teb_local_planner.

[26] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[27] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[28] Z. Wang, Y. Wu, and Q. Niu, "Multi-sensor fusion in automated driving: A survey," *Ieee Access*, vol. 8, pp. 2847–2868, 2019.

[29] J. Zhang, S. Singh *et al.*, "LOAM: Lidar odometry and mapping in real-time." in *Robotics: Science and systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[30] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[31] R. P. Maintainers, "depthimage_to_laserscan," https://wiki.ros.org/depthimage_to_laserscan.

[32] ——, "pointcloud_to_laserscan," https://github.com/ros-perception/pointcloud_to_laserscan.

[33] O. Noel, "Depth point filter-converter to Scan," https://github.com/Noceo200/depth-filter-scan-converter.

[34] J. Bloomenthal and J. Rokne, "Homogeneous coordinates," *The Visual Computer*, vol. 11, pp. 15–26, 1994.

[35] M. Kam, X. Zhu, and P. Kalata, "Sensor fusion for mobile robot navigation," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 108–119, 1997.

[36] V. E. Kremer, "Quaternions and SLERP," in *Embots. dfki. de/doc/seminar ca/Kremer Quaternions. pdf*, 2008.