Efficient Evaluation of SLAM Methods and Integration of Human Detection with YOLO Based on Multiple Optimization in ROS2

Hoang Tran Ngoc*, Nghi Nguyen Vinh, Nguyen Trung Nguyen, Luyl-Da Quach
FPT University, Can Tho 94000, VietNam

***Abstract*—In the realm of robotics, indoor robotics is an increasingly prominent field, and enhancing robot performance stands out as a crucial concern. This research undertakes a comparative analysis of various Simultaneous Localization and Mapping (SLAM) algorithms with the overarching objective of augmenting the navigational capabilities of robots. This is accomplished within an open-source framework known as the Robotic Operating System (ROS2) in conjunction with additional software components such as RVIZ and Gazebo. The central aim of this study is to identify the most efficient SLAM approach by evaluating map accuracy and the time it takes for a robot model to reach its destinations when employing three distinct SLAM algorithms: GMapping, Cartographer SLAM, and SLAM_toolbox. Furthermore, this study addresses indoor human detection and tracking assignments, in which we evaluate the effectiveness of YOLOv5, YOLOv6, YOLOv7, and YOLOv8 models in conjunction with various optimization algorithms, including SGD, AdamW, and AMSGrad. The study concludes that YOLOv8 with SGD optimization yields the most favorable outcomes for human detection. These proposed systems are rigorously validated through experimentation, utilizing a simulated Gazebo environment within the Robot Operating System 2 (ROS2).**

Keywords—Indoor robotic; SLAM; ROS2; Robot model; Human detection; YOLO

I. Introduction

With the rapid advancement of Artificial Intelligence (AI) and the continuous evolution of sensor technologies, coupled with the introduction of the Robot Operating System (ROS) and its latest iteration, ROS 2 [1], the development of indoor robots has become more accessible than ever before. Mobile robots have yielded substantial economic benefits across various sectors, including industry, warehousing, and logistics [2]. Notably, robots are no longer confined to industrial applications; they are increasingly being deployed in the realm of mental healthcare, where they assist therapists in caring for the elderly and children struggling with depression [3]. In light of these developments, our objective is to create a pet robot designed to serve the purposes mentioned above. Within the realm of developing such a robot, we consider two fundamental tasks of paramount importance: Navigation and Human Detection. This article aims to compare different methods and put forth the most optimal approaches for accomplishing these critical tasks.

Sensors play a crucial role in how robots and autonomous vehicles perceive their surroundings. The choice and installation of sensors have a significant impact on the specific results of observation and also influence the complexity of SLAM problems. Based on the primary type of sensor used, SLAM can be categorized into Visual-SLAM and LiDAR-SLAM. Visual SLAM remains a particularly challenging task due to inherent difficulties. Moreover, vision cameras struggle to extract features from texture less areas, which limits the applicability of Visual SLAM. On the other hand, LiDAR SLAM primarily relies on LiDAR technology for environmental sensing. The relative movement and pose changes of the laser radar are determined by comparing point clouds captured at different moments. LiDAR SLAM offers advantages in terms of stability, simplicity, precise map data, and lower computational requirements compared to Visual SLAM. Numerous investigations have explored LIDAR-SLAM techniques in the literature [4]-[9].

This research primarily centers on evaluating Slam techniques through the utilization of ROS2, with a primary emphasis on assessing their performance based on the resulting maps. The evaluated methods fall within the category of 2D LIDAR Slam techniques designed for indoor settings, including GMapping, Cartographer SLAM, and Slam-Toolbox. In addition, we have integrated a LIDAR-SLAM technique with a human detection system. The methods we evaluated and tested for this integration include YOLOv5 [10], YOLOv6 [11], YOLOv7 [12] and YOLOv8 [13], using multiple collected datasets. Ultimately, this research aims to develop an optimized system for indoor robotic operations, ensuring precise localization and robust human detection capabilities within indoor spaces. This system is designed to facilitate avoidance maneuvers and ensure safety during robot operations.

II. Related Work and Our System

A. Related Work

Recent advancements in the field of mobile robot navigation have enabled robots to operate effectively in various environments, including warehouses, retail stores, and crowded pedestrian areas. A plethora of navigation solutions have been proposed to address these challenges [14]. One of these solutions introduced a navigation system and environmental representation that utilized 3D data obtained from tilting 2D laser scanners for navigation. Initially, conventional methods such as A* and DWA were employed

*Corresponding Author.

for path planning and obstacle avoidance. However, the prevalence of cost-effective 3D depth cameras and laser scanners has gradually overshadowed the use of tilting 2D laser scanners. Some aspects of these approaches were customized to account for the specific geometry, limited accuracy, and characteristics of 2D tilting laser setups. Furthermore, the emergence of sparse multi-beam laser scanners has rendered traditional raycasting methods ineffective in clearing free space, especially in dynamic environments. In response to this challenge, Sparse Traversability Volume (STVL) has emerged as a scalable alternative suitable for various types of sparse and long-range sensors, replacing traditional techniques effectively [15].

The Robot Operating System (ROS) Navigation has historically been one of the most popular navigation solutions built on top of ROS. However, with the introduction of ROS2, Navigation2 was developed as a successor to build upon the success of ROS Navigation. Navigation2 incorporates a behavior tree for orchestrating navigation tasks and utilizes novel methods designed to handle dynamic environments, making it applicable to a wider range of modern sensors. SLAM is a critical technology in mobile robotics, allowing robots to map unknown environments while simultaneously determining their own position based on the created map. Several SLAM algorithms have been proposed, categorized into two groups: earlier algorithms employing Bayes-based filter approaches like GMapping [16], and newer ones using graph-based methods such as Cartographer [17], Karto SLAM [18], and Slam Toolbox [19].

For our human detection task, we have chosen to leverage the YOLO (You Only Look Once) framework for several compelling reasons. Among the numerous object detection algorithms available, YOLO stands out due to its exceptional combination of speed and accuracy. It excels in rapidly and accurately identifying objects within images, making it an ideal choice for real-time applications. YOLO has gained prominence as a central system for real-time object detection in various domains, including robotics, autonomous vehicles, and video monitoring, demonstrating its reliability and versatility in a wide range of applications [20]. The YOLO family of object detection models has undergone a series of iterations, evolving from the original YOLOv1 to the most recent YOLOv8. Each iteration has built upon the foundation of its predecessors, aiming to address limitations and enhance overall performance in object detection tasks. However, in our research paper, we will specifically focus on evaluating human detection using YOLO versions ranging from YOLOv5 to YOLOv8. This selective approach allows us to assess the advancements and capabilities of these more recent YOLO iterations in the context of human detection, which is a pivotal aspect of our study.

B. Our System

In this study, we are using an open-source robotic middleware ROS 2. Inheriting from ROS, the libraries provided by ROS 2 used in this project make robotic programs and development more flexible and easier. ROS 2 also offers the capability to obtain hardware abstractions of the robot model, encompassing sensors, motors, and actuators which can be used for navigation and are accessible through URDF and XACRO files. Additionally, we utilize other open-source software in conjunction with ROS 2, namely Gazebo and RVIZ 2. Gazebo functions as simulation software, enabling the virtual simulation of robots through the use of plugins. This allows us to construct a simulated environment represented as URDF and WORLD files, facilitating the simulation of the robot within the Gazebo world, as depicted in Fig. 1. In this article, the robot model used is a mobile robot with two wheels, equipped with several sensors including a 360° 2D LIDAR and a camera. These sensors play a crucial role in tasks like navigation and human tracking. The detailed illustration of this robot model is shown in Fig. 2. RVIZ 2 serves as a data visualizer tool for robot data, presenting various data types, including laser scans, maps retrieved from the map server, and grid displays. As well as ROS, RVIZ 2 also provides navigation goals and poses estimation functionalities, which are inherited from RVIZ in order to accomplish the Robot's navigation in the ROS 2 virtual environment.

In order to make the autonomous robot navigate in its environment safely without encountering collisions with either static or moving obstacles, the assistance of SLAM is required. Different SLAM algorithms can be used for mapping such as Karto SLAM, Cartographer SLAM, Gmapping SLAM, and Hector SLAM, but in this paper, we want to compare the package provided by ROS 2 (SLAM_toolbox).

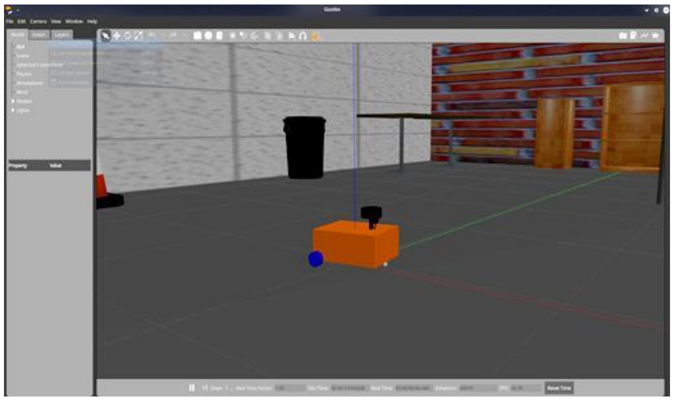


Fig. 1. Robot model in Gazebo virtual world.

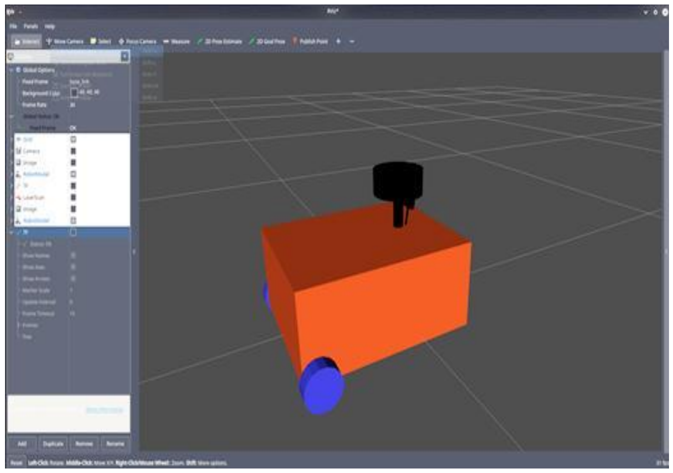


Fig. 2. Robot description on Rviz 2.

III. Methodology

A. LIDAR 2D SLAM Algorithms

In this section, an analysis of SLAM algorithms in the ROS2 environment will be conducted. Among the available open-source laser scanner SLAM algorithms, Cartographer is a graph-based algorithm that manages a graph representing robot poses and features. It offers resource efficiency, especially for constructing large-scale maps. It consists of a front-end responsible for tasks like scan matching, trajectory building, and submap generation, as well as a back-end that handles loop closure procedures, using the Google Ceres graph solver. Cartographer provides a pure localization mode for users with existing maps and supports data serialization for storing processed sub-maps. However, it has encountered challenges, such as the discontinuation of maintenance and support by Google, leading to its abandonment. This algorithm divides a large single map into smaller sub-maps by integrating two separate 2D SLAM techniques. One method focuses on local operations, while the other deals with global aspects, and both employ a LiDAR sensor. These two methods are optimized independently. In the local SLAM process, sub-maps are created by collecting and arranging data, involving the alignment of multiple scans relative to the initial position. These sub-maps are grids with defined resolutions, with each grid point indicating occupancy probability based on prior measurements updated as new sub-maps are generated. An algorithm optimizes sub-map positioning for alignment, aiding extrapolation. The second part, global SLAM, leverages feedback from these sub-maps, which are associated with robot positions. This enhances maps and reduces accumulated SLAM errors, a process known as loop closure [24]. The well-known optimization technique called Spare Pose Adjustment (SPA) [23] [38] is utilized in Cartographer SLAM, and a map-scanner is activated whenever a sub-map is generated to close the loop and incorporate that sub-map into the graphic. Two formulas are provided to determine whether a cell is classified as busy, empty, or transitioning to an empty state within a map cell, enhancing comprehension.

$$M_{new}(\text{cell}) = F^{-1}(F(M_{old}(\text{cell})\ F(f_{hit}))) \tag{1}$$

where: $M_{old}(cell)$ is the old probability of the cell which could be an error, f_{hit} is the probability function that represents a map cell is busy, and $F = \frac{F}{1-F}$.

Scan matching process goes through a minimization of the following function:

$$\arg_{\delta}\min \sum_{k=1}^{K} (1\text{-}M_{smoothen}(T_{\delta} s_k))^2 \tag{2}$$

where, $M_{smoothen}$ represents the value of a cell that has been smoothed using its neighboring values, s_k denotes the laser scan reading involves to the cell, T_δ is the matrix transformation that displaces the point h_k to δ, and δ is the posture vector $(\delta_x, \delta_y, \delta_\theta)$.

Additionally, Cartographer may struggle to create suitable maps for annotation and localization when integrated with other robotic platform localization software that lacks exceptional odometry. Its complexity can hinder modifications and resolution of seemingly straightforward issues, limiting its suitability for many applications.

GMapping Slam is widely used within the Robot Operating System (ROS) and stands out for its frequent adoption. GMapping employs the Rao Blackwellized Particle Filter (RBPF) [21] technique for map generation. However, it's important to note that GMapping has limitations when applied to large environments and struggles with precise loop closure in industrial-scale spaces. The idea of the RBPF for SLAM, first introduced by Murphy [22] in 1999, is to estimate the joint posterior $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$ of the map where m is the map and the trajectory $a_{1:t} = a_1, \dots, a_t$ of the robot. The $z_{1:t} = z_1, \dots, z_t$ is the given observations and the odometry measurements is $o_{1:t-1} = o_1, \dots, o_{t-1}$. These both can be obtained by the robot's data. The fully RBPF factorization for SLAM is representing below:

$$p(m, a_{1:t} \mid z_{1:t}, o_{1:t-1}) = p(m \mid a_{1:t}, z_{1:t})\ p(a_{1:t} \mid z_{1:t}, o_{1:t-1}) \tag{3}$$

Nonetheless, filter-based approaches such as GMapping encounter difficulties when attempting to achieve seamless reinitialization across multiple sessions.

The SLAM Toolbox is a versatile mapping solution that efficiently covers large areas using standard mobile Intel CPUs commonly found on robots. It simplifies space mapping through automation and supports session serialization, enabling users to easily improve existing maps. What makes it unique is its preservation of complete raw data and pose-graph, enabling various innovative tools like manual pose-graph manipulation and kinematic map merging. The SLAM Toolbox provides three primary operational modes: synchronous mapping for high-quality maps, asynchronous mapping for real-time performance, and pure localization for adapting to dynamic environments.

Significant enhancements to the OpenKarto SLAM library [19], [38] have boosted its speed and adaptability, making it a valuable tool for robot navigation and mapping tasks. The SLAM Toolbox has been effectively integrated, tested, and utilized on diverse robotic platforms around the world by both professionals in the industry and researchers. It serves as the default SLAM solution in ROS 2, replacing GMapping. Incorporated into the ROS 2 Navigation2 project, it enables real-time positioning in changing environments, facilitating autonomous navigation. Its user-friendly interface empowers both experts and non-experts to map extensive spaces in real time, establishing its significance in robotics and autonomous system development.

To evaluate a Slam algorithm, two criteria such as the accuracy of the generated map based on comparing the ground truth and the map generated or its feature locations, and the time taken for the robot to navigate and get to its destination for each environment are considered. In order to investigate the map's quality, we created three different simulation environments on the Gazebo simulator, and we added the robot model to those simulation worlds that we created. The robot model has laser scanner sensors (LIDAR 360) that supply data crucial for map creation when implementing three

SLAM techniques in RVIZ 2. The data were transmitted to the scan topic for the map, to generate and then visualize it on RVIZ 2; afterward, map topics were activated. The higher the map fits the ground truth the better the map quality. Fig. 3 shows RVIZ 2 and Gazebo when launching our virtual world from their respective directories. In Fig. 4, we illustrate three maps generated by three different SLAM methods within a simulated environment.

These maps are saved in PGM file format and have been compared with ground truth data. To create these maps, the robot navigated through the simulated environment using a control framework called "ros2_control," which is a reimplementation of the "ros_control" framework used in ROS.

After mapping each environment using different SLAM techniques with their default parameters, the resulting maps were saved as YAML files. Notably, we observed that the SLAM_toolbox method exhibited higher accuracy. The map generated by this SLAM method had less noise and better alignment with the ground truth. Detailed evaluations of these SLAM methods will be discussed further in the experimental section.

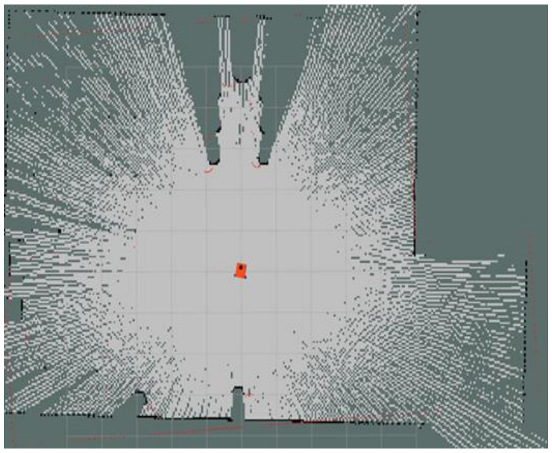


Fig. 3. Robot model creating the map of the environment.

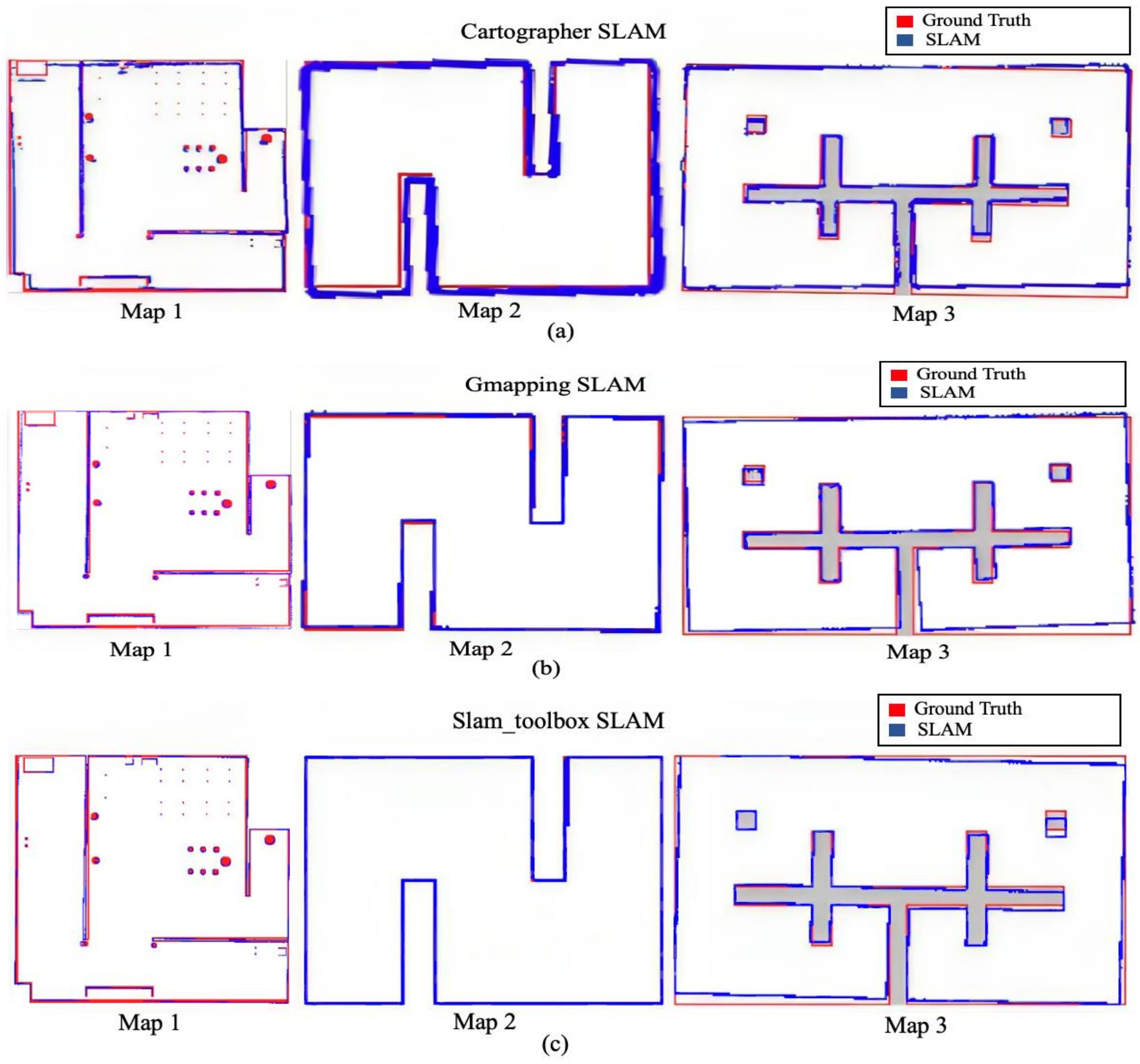


Fig. 4. Generated map results compared to ground truth with: (a) Cartographer SLAM, (b) Gmapping SLAM, (c) Slam_toolbox SLAM.

To enable autonomous navigation of the robot, the robot received a generated map of each algorithm as an input. The AMCL (Adaptive Monte Carlo Localization) algorithm employs a probabilistic localization model that aids the movement of the robot between different positions using a "2D Navigation Goal" provided by RVIZ 2 to reach each destination; and "2D Pose Estimator" that allows the position of the robot model to be set within an environment.

B. Human Detection Algorithm

In this section, we scrutinize and assess the human detection algorithm, aiming to propose an optimal model for indoor robots. This model is designed to help robots in tasks like obstacle avoidance and ensuring operational safety. Among the many algorithms for object detection, the YOLO (You Only Look Once) framework has gained recognition for its exceptional blend of speed and accuracy, enabling rapid and reliable object identification in images. Over time, the YOLO family has gone through multiple iterations, with each new version building upon the previous ones to address shortcomings and enhance performance.

In 2020, Ultralytics introduced YOLOv5 [10]. Unlike YOLOv4, which utilized Darknet, YOLOv5 was developed using Pytorch and incorporated various improvements. YOLOv5 also integrated an AutoAnchor algorithm, a pre-training tool that evaluates and adjusts anchor boxes to better suit the dataset and training parameters, including image size. Initially, it applies a k-means function to dataset labels to establish starting conditions for a Genetic Evolution (GE) algorithm.

In September 2022, the Meituan Vision AI Department introduced YOLOv6 [11]. Its network design features an efficient backbone PAN topology neck, which utilizes RepVGG or CSPStackRep blocks. It uses an efficient decoupled head with a hybrid-channel strategy. Furthermore, a new quantization technique was proposed to achieve faster and more accuracy.

YOLOv7 [12] was published in July 2022. At the time of its release, it outperformed all known object detectors in terms of speed and accuracy, achieving frame rates ranging from 5-160 FPS. It was trained only on the MS COCO dataset without pre-trained backbones. YOLOv7 brought forth numerous architectural modifications and a range of improvements, which boosted accuracy without affecting inference speed, albeit increasing training time.

In January 2023, Ultralytics, the organization responsible for YOLOv5, released YOLOv8 [13]. It has the capability to handle multiple vision-related tasks, including classification, object detection, segmentation, pose estimation, and tracking.

In Table I, we compare the structure and loss functions employed in various versions of YOLO. Fig. 5 illustrates the structure of the human detection model using YOLOv8-N, which we propose for adoption. With its compact size and high accuracy, YOLOv8-N is well-suited for deployment in resource-constrained hardware robots. Our evaluations indicate that YOLOv8-N outperforms other models. The results of our indoor detection capabilities will be presented in the experimental section.

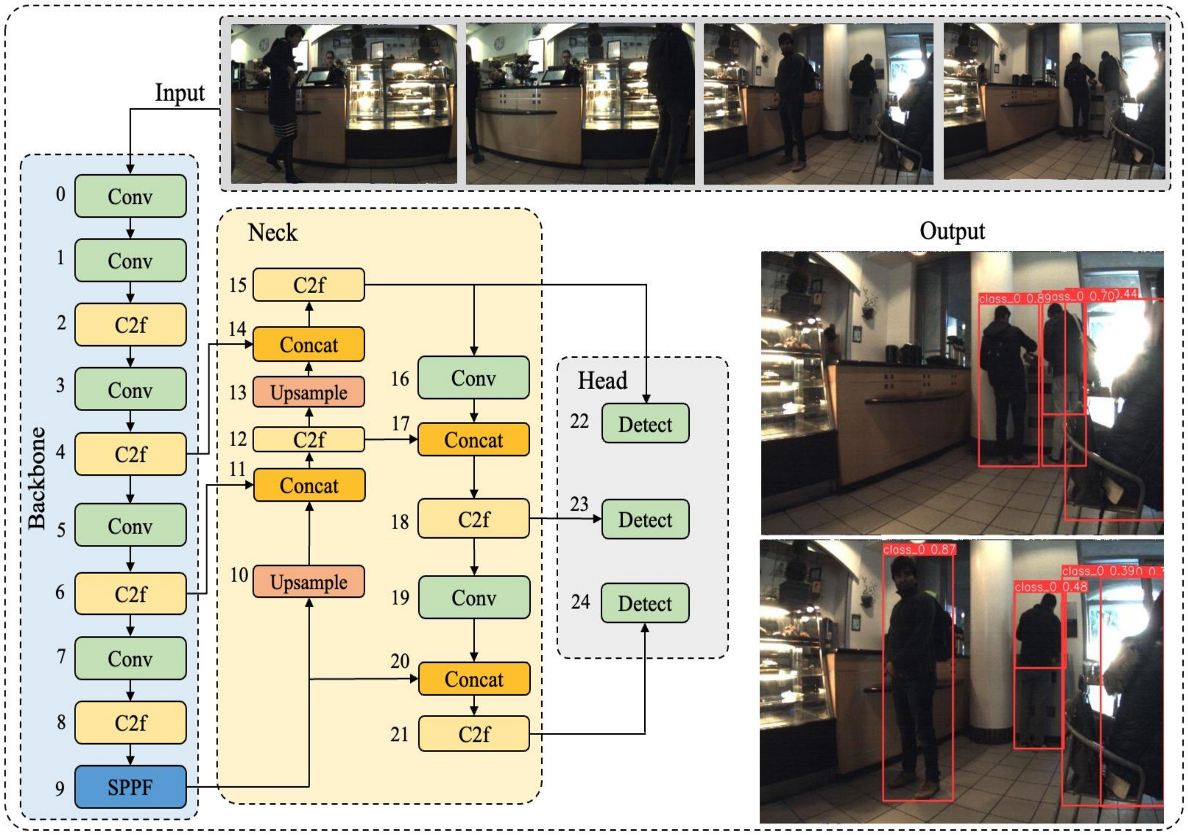


Fig. 5. The structure of the human detection model using YOLOv8-N.

TABLE I. ARCHITECTURE OF YOLO'S VERSION

	YOLOv5-N	YOLOv6-N	YOLOv7-Tiny	YOLOv8-N
Backbone	CSP Darkent53 [30]	RepVGG [31] and CSPRepStack [32]	EELAN [12]	CSPDarkent53 [30]
Neck	PANet	RepPAN	PANet	PAN-FPN
Head	B x (5 +C)	Decoupled Classification, and Detection Head	Lead Head	Decoupled Head
Loss Function	Binary Cross Entropy (BCE) [25], and Logit Loss Function (LLF)[26]	Varifocal Loss (VFL) [26], and Distribution Focal Loss (DFL) [27]	BCE with Focal Loss, and IoU [28]	VFL, DFL Loss, and CIOU [29]
Parameters	1.9M	4.3M	6.2M	3.2M

IV. EXPERIMENTAL RESULTS

A. SLAM Performance and Results

After utilizing the three SLAM techniques, Cartographer, GMapping, and SLAM_toolbox, the map was successfully generated with the highest level of precision using the SLAM_toolbox method, as depicted in Fig. 4. To assess the impact of map accuracy, we recorded the time taken by the robot to navigate and reach its destination in various environments. We divided the time measurement process into three segments for each map, conducted multiple test runs, and subsequently computed the average duration.

The first two phases of testing were conducted on Map_2 and Map_3, as illustrated in Fig. 6. In each of these navigation scenarios, a map generated by one of the three SLAM algorithms was utilized. Table II presents the time taken by the robot to reach its destination on the maps generated by Cartographer SLAM, GMapping, and SLAM_toolbox. For map 2, the goal point coordinates were set as (x = 4.0, y = 2.0), while for Map_3, they were set as (x = 0.0, y = 6.0), with the z-axis being maintained at 0.

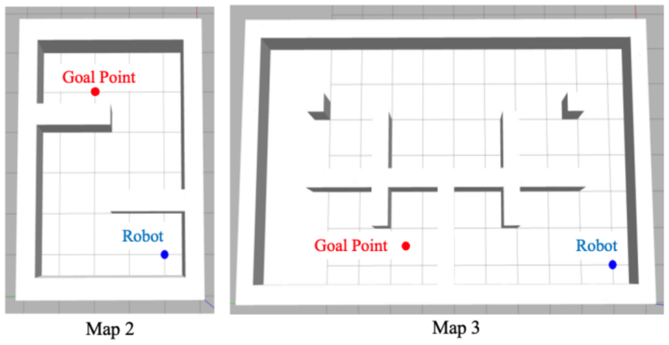


Fig. 6. Map 2 and 3 with their goal point.

TABLE II. THREE SLAM METHOD TRAILS FOR THE SECOND AND THIRD MAP

	Cartographer (s)		Gmapping (s)		Slam_toolbox (s)	
	Map_2	Map_3	Map_2	Map_3	Map_2	Map_3
Test 1	25.90	72.64	25.73	70.72	**24.29**	**70.11**
Test 2	26.10	70.15	26.10	72.17	**24.45**	**70.12**
Test 3	26.24	72.45	26.12	72.66	**24.71**	**70.07**
Average	26.08	71.75	25.983	71.85	**24.483**	**70.1**

Upon examination of Table II, it becomes apparent that when the robot is in operation with the SLAM_toolbox map, it achieves a quicker trajectory completion compared to the two other methods. This phenomenon is attributed to the SLAM_toolbox's capacity to produce maps characterized by a higher degree of precision and reduced noise levels. Consequently, the robot's operational stability is significantly improved.

A wide and intricate map with numerous obstacles was tested, as shown in Map_1. Fig. 7 illustrates the map when the robot navigates to its destination, with the green line indicating the path that the robot model must follow to reach its destination. The destination is set using RVIZ's 2D Goal Pose tool and is represented by a green arrow.

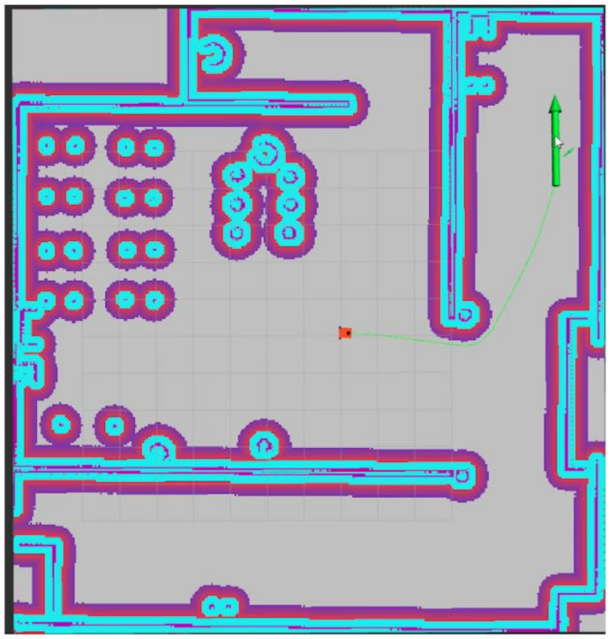


Fig. 7. Robot navigates to its destination in Map_1.

The last part was tested on Map_1 with three goal points. Each of these navigation tasks is executed using a map generated by its respective SLAM algorithm. The coordinates for point 1 is (x = 7.5, y = -2.0, z = 0.0); point 2: (x = 4.0, y = -8.0, z = 0.0); point 3: (x = -5.0, y = 0.0, z = 0.0). Fig. 8 shows Map_1 and the three goal points where the robot will move to, each represented by specific coordinates.

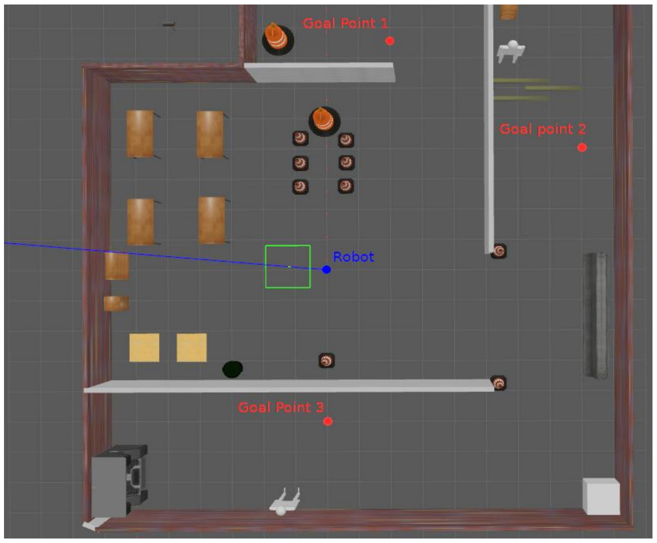


Fig. 8. The Map_1 with 3 different goal points.

Table III presents the duration it took for the robot to reach its destinations on the maps created by each method. It was observed that the map created using Slam Toolbox is more accurate than GMapping and Cartographer SLAM. To calculate the accuracy percentage of the point on the created map compared to the point we give before, we use the following Eq. (4):

$$Accuracy\ (\%) = [1 - (\frac{Error}{Range})] * 100 \quad (4)$$

where: Error is the Euclidean distance between actual goal points and the robot location after navigating, Range is the maximum possible distance between the two points, which is the distance between the actual point and the origin (0, 0) on Gazebo virtual world. Although the travel times to the destination points being relatively close for the use of maps in all three methods, Table IV shows the robot movement of the SLAM_toolbox method is more accurate to the virtual world than the other. The reason is Gmapping uses a particle filter to build grid maps from 2D lidar data and primarily relies on LIDAR data for mapping which is not well suited for expansive environments and struggles to accurately complete loop closures on an industrial scale, and the Cartographer relies on incorporating LIDAR and IMU data for highly accurate mapping because of its complexity in algorithm (IMU is not used in this robot model). Based on the empirical evidence from our conducted experiments, we readily infer that the utilization of the SLAM_toolbox for SLAM mapping is demonstrably superior.

B. Human Detection Results

The experiments were conducted to train and evaluate YOLO's versions using two distinct datasets. The training model was executed on an Ubuntu 20.04 platform, utilizing an Intel Core i7 processor and 16 GB of RAM. Firstly, we train YOLO's versions 20 epochs with the same hyparameters on a small dataset to choosing up the best optimal version for our work. After that we train chosen version 50 epochs on larger dataset with different optimizations such as SGD, AdamW [33], and AMSGrad [34]. Performance was assessed based on metrics such as Recision, Recall, and mAP(0.5-0.95).

1) Datasets and Evaluation Metrics:

a) Datasets: The research paper ultilizes three datasets created by using framework FiftyOne [35] to spliting human images, bounding box from the MSCOCO dataset [36], and JRDB dataset [39]. The small dataset contains 12.6K images with 10K for train and 2.6K for validation. The large dataset ontains 66.6K images with 64K for train and 2.6K for validation. Moreover, we also use augmentation methods on each of those dataset before training. Augment methods used are Blur, MedianBlur, ToGray, CLAHE.

TABLE III. THREE SLAM METHOD TIME RECORDED FOR THE FIRST MAP

	Cartographer (s)			**Gmapping (s)**			**Slam Toolbox (s)**		
	1st destination	2nd destination	3rd destination	1st destination	2nd destination	3rd destination	1st destination	2nd destination	3rd destination
Test 1	35.76	41.32	55.31	35.98	43.39	54.29	**35.05**	**41.35**	**53.58**
Test 2	35.90	42.35	54.91	35.74	43.51	54.77	**35.22**	**41.62**	**53.79**
Test 3	35.04	42.38	55.39	35.20	43.28	54.42	**35.01**	**41.84**	**53.45**
Average	35.56	42.01	55.20	35.64	43.39	54.49	**35.09**	**41.60**	**53.60**

TABLE IV. DISTANCE ACCURACY OF THREE SLAM METHODS FOR THE FIRST MAP

	Cartographer (m)			**Gmapping (m)**			**Slam Toolbox (m)**		
	Goal point 1	Goal point 2	Goal point 3	Goal point 1	Goal point 2	Goal point 3	Goal point 1	Goal point 2	Goal point 3
Test 1	x: 7.73867 y: -1.93425	x: 4.11695 y: -8.24249	x: -4.9892 y: -0.4127	x: 7.4873 y: -2.1837	x: 4.0047 y: -7.838	x: -4.988 y: 0.210	**x: 7.583 y: -2.088**	**x: 4.036 y: -8.046**	**x: -5.007 y: -0.027**
Test 2	x: 7.73895 y: -1.93484	x: 4.11672 y: -8.2428	x: -4.9896 y: -0.4125	x: 7.4933 y: -2.1807	x: 4.0123 y: -7.8801	x: -4.874 y: 0.1199	**x: 7.484 y: -2.018**	**x: 4.06252 y: -8.13627**	**x: -5.0832 y: -0.0174**
Test 3	x: 7.7385 y: -1.9343	x: 4.1163 y: -8.24216	x: -4.9890 y: -0.4122	x: 7.4854 y: -2.1858	x: 4.0098 y: -7.8823	x: -5.003 y: -0.173	**x: 7.5276 y: -2.092**	**x: 4.04852 y: -8.07527**	**x: -4.968 y: -0.027**
Average	x: 7.7387 y: -1.9344	x: 4.11668 y: -8.24248	x: -4.9893 y: -0.41254	x: 7.48874 y: -2.1834	x: 4.0089 y: -7.8668	x: -5.1035 y: -0.173	**x: 7.532 y: -2.066**	**x: 4.04952 y: -8.08527**	**x: -5.0198 y: -0.0242**
Accuracy	96.81%	97.0%	91.74%	97.63%	98.5%	96.04%	**99.05%**	**98.89%**	**99.37%**

b) Evaluation metrics: In object detection challenges and scientific research, various annotated datasets are used, and the primary metric for assessing the accuracy of object detections is the Average Precision (AP). AP relies on four fundamental concepts:

- True positive (TP): Accurately identifying a real bounding box in line with the ground truth.
- False positive (FP): Incorrectly identifying a non-existent object or inaccurately placing an existing object's detection.
- False negative (FN): Failing to detect a bounding box that matches the ground truth.
- True Negative (TN): Nevertheless, in the realm of object detection, the concept of True Negative (TN) is not relevant.

To address this, the intersection over union (IOU) metric is commonly employed. IOU is a measurement based on the Jaccard Index, which quantifies the similarity between two sets of data. In object detection, IOU calculates the overlap between the predicted bounding box (Bp) and the ground-truth bounding box (Bgt), dividing it by the area of their union. This provides a more suitable and informative metric for evaluating object detection performance, particularly in scenarios where True Negatives are not relevant [37].

In our research, we will primarily evaluate the performance of our model using the mean Average Precision (mAP) metric over a range of IoU thresholds (0.5 to 0.95). This mAP score considers the integral of mAP across these different IoU thresholds, providing a comprehensive assessment of detection accuracy. Additionally, we will also utilize the Precision metric to evaluate the percentage of correct positive predictions made by our model and the Recall metric to assess the percentage of correct positive predictions among all the ground truth annotations. These metrics collectively offer a well-rounded evaluation of our model's performance in object detection tasks.

The equation of those metrics is shown below:

$$\text{IoU} = \frac{\text{area of overlap}}{\text{area of union}} \tag{5}$$

$$\text{Precision} = \frac{\text{TP}}{\text{FP+TP}} \tag{6}$$

$$\text{Recall} = \frac{\text{TP}}{\text{FN+TP}} \tag{7}$$

2) Results and discussion: Fig. 9 to Fig. 11 illustrate the mAP (0.5-0.95), precision, and recall, while Table V presents the final results of training models on our dataset. In this set of results, two models, YOLOv8 and YOLOv6, outperform YOLOv5 and YOLOv7, with mAP@0.5:0.95 scores of 0.438 and 0.503, compared to 0.424 and 0.407, respectively. Notably, YOLOv6 demonstrates superiority across these metrics, as evidenced by its leading performance in Fig. 9 with the highest mAP score.

However, despite YOLOv6's impressive performance, we decided to exclude it from further consideration due to two critical factors. Firstly, it boasts a substantial number of parameters, totaling 4.3 million, which can strain computational resources. Secondly, its convergence appears to plateau within the first 20 epochs, suggesting that other models might potentially achieve even higher performance with prolonged training. Thus, we have opted for a trade-off between processing speed and accuracy in our model selection process.

Furthermore, YOLOv8, while not particularly outstanding in the initial 20 epochs, is our preferred choice. This decision is rooted in its consistent convergence, indicating that with more training time, it can potentially outperform other models. Additionally, as referenced in [13], YOLOv8 currently boasts superior processing speed compared to its counterparts, making it well-suited for real-time object detection applications.

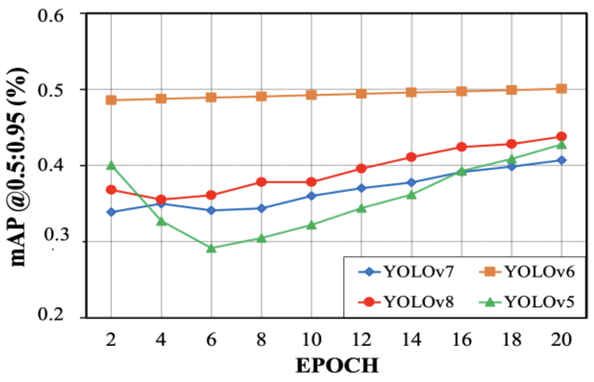


Fig. 9. mAP@0.5:0.95 of YOLO's versions.

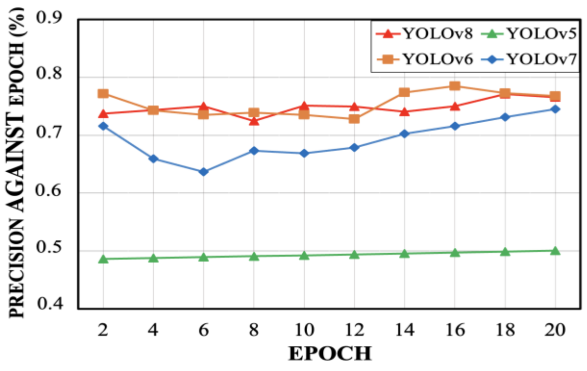


Fig. 10. Precision of YOLO's versions.

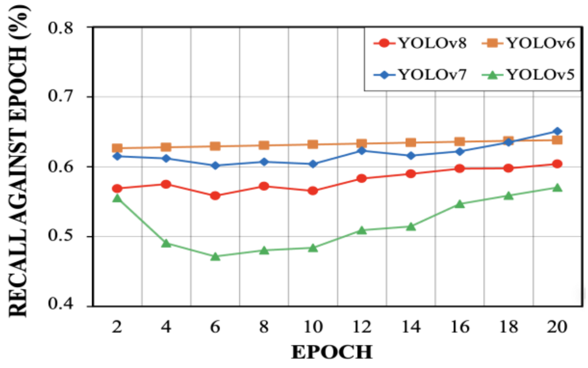


Fig. 11. Recall of YOLO's versions.

TABLE V. LAST RESULT OF FIRST EXPERIMENT

	Precision	Recall	AP (IoU=0.5)	AP (IoU=0.5:0.95)
YOLOv5	0.503	0.58	0.696	0.424
YOLOv6	0.768	0.651	0.791	0.503
YOLOv7	0.725	0.679	0.664	0.407
YOLOv8	0.766	0.604	0.728	0.438

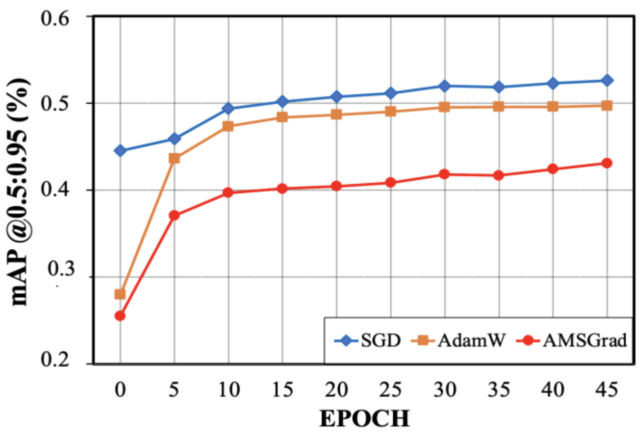


Fig. 12. mAP@0.5:0.95 of YOLOv8 with optimizers.

In the second experiment, we conducted training for 50 epochs using YOLOv8 on a dataset larger than the first with 66.6K images, employing various optimizers, including SGD, AdamW, and AMSGrad. The results from the last epoch are presented in Table VI. Notably, the SGD optimizer demonstrated superior performance across precision, recall, and mean Average Precision (mAP) with 0.794 Precision, 0.657 Recall, and 0.527_mAP@0.5:0.95. It has shown an increase over the original YOLOv8 model in [13] with only smaller than 0.4 mAP@0.5:0.95, which is obvious because here the parameters are trained to focus exclusively on human detection compared to numbers of class up to 80 in the original model. In Fig. 12, 13, and 14, we provide insights into the mAP@0.5:0.95, precision, and recall results throughout the 50-epoch training process. Remarkably, SGD consistently outperformed the other optimizers. Nonetheless, it's worth highlighting that even at the 50th epoch, all three methods exhibited ongoing improvement. This suggests that they have not yet reached their maximum potential, indicating that AdamW and AMSGrad might still have untapped strengths.

In Fig. 15, we conducted another evaluation to assess the human detection capability of the YOLOv8-SGD framework that we propose for the task of detecting individuals, particularly humans, within the JRDB dataset. This evaluation involved detecting individuals in various indoor settings, such as shopping centers, train stations, and cinemas.

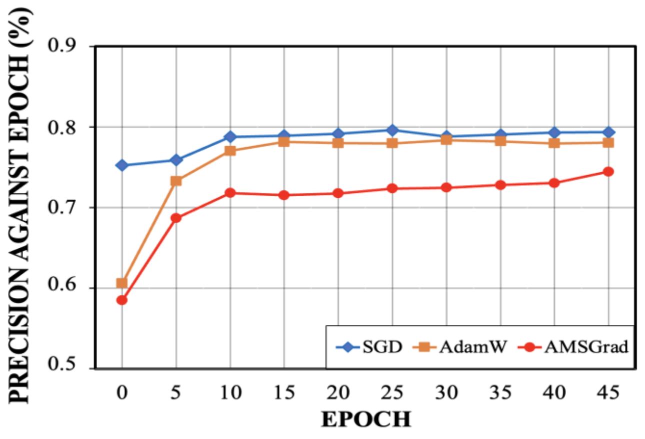


Fig. 13. Precision of YOLOv8 with optimizers.

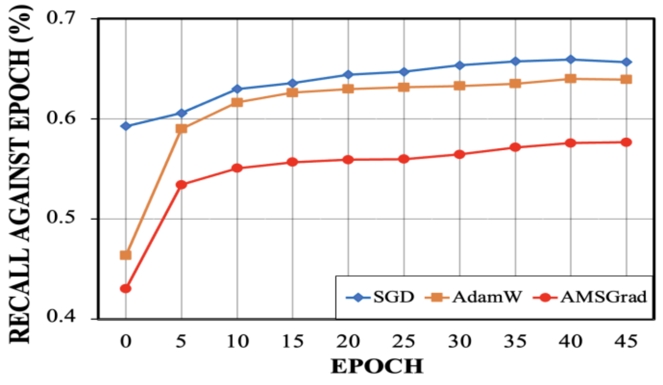
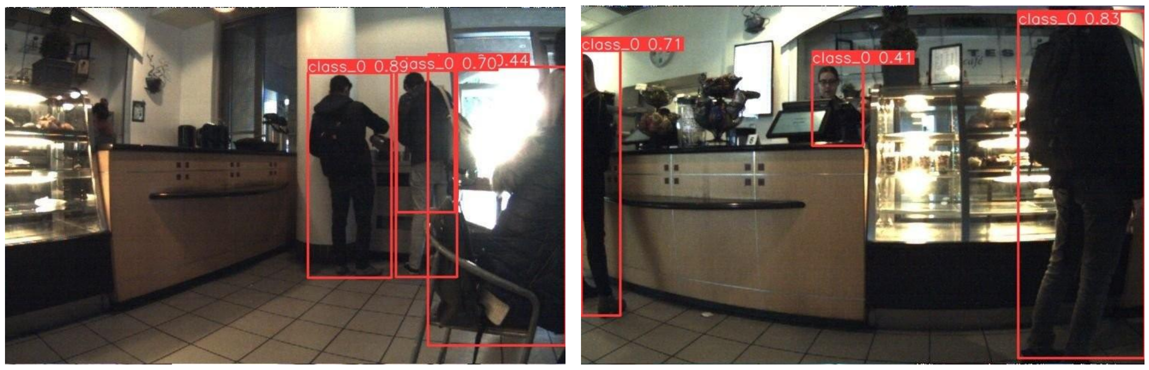


Fig. 14. Recall of YOLOv8 with optimizers.

TABLE VI. LAST RESULT OF THE SECOND EXPERIMENT

	Train /Box _loss	**Val /Box _loss**	**Val /dfl _loss**	**(Pre)**	**(Re)**	**AP IoU =0.5**	**AP IoU= 0.5:0.95**
YOLO v8-SGD	1.024	1.029	1.08	0.794	0.657	0.757	0.527
YOLO v8-ADAMW	1.082	1.087	1.15	0.782	0.639	0.735	0.498
YOLO v8-AMSGrad	1.205	1.2	1.28	0.754	0.577	0.674	0.438



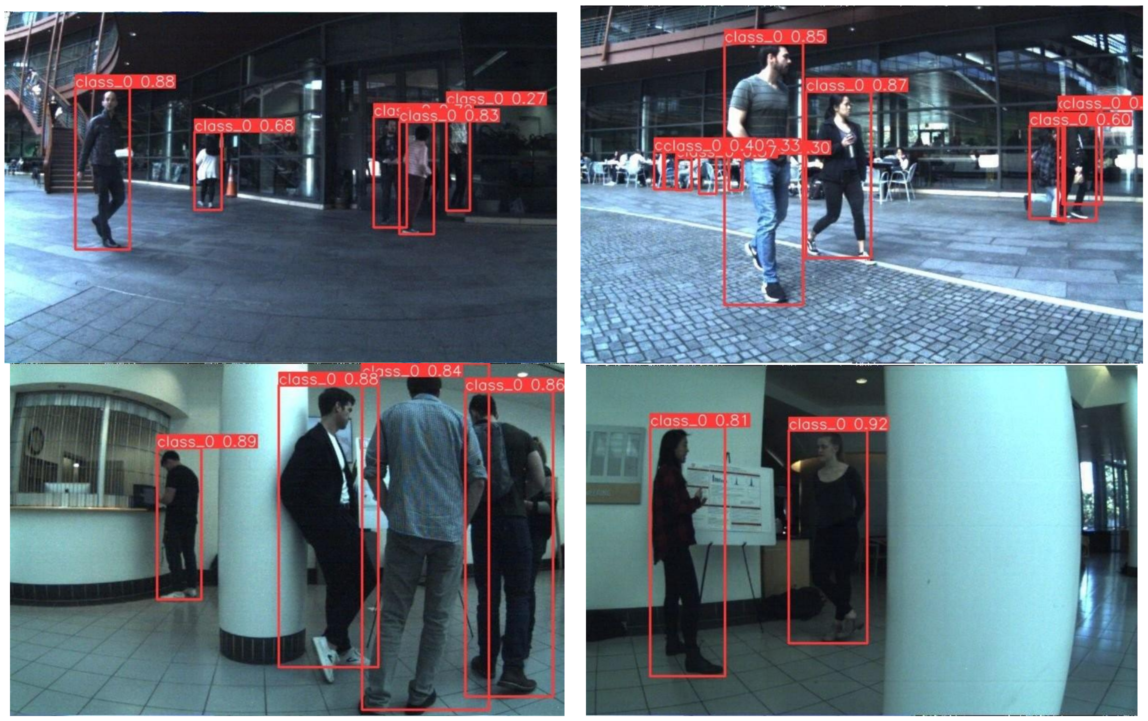


Fig. 15. The human detection capability of the YOLOv8-SGD framework with JRDB dataset.

V. Conclusion

We have successfully compared currently commonly used state-of-the-art methods for each task such as SLAM, Navigation, and Human detection. Furthermore, we have also provided analysis and evaluation of the experimental results.

Our findings align with our expectations, and we have identified potential areas for enhancement. We utilized the maps generated by each algorithm in both RVIZ 2 and Gazebo for guiding the robot to three distinct destinations. We repeated this process in five trials. The average values obtained from these tests were then used to create graphs that depict the time taken for various destinations, as shown in the table. The comparison revealed that the map generated with the Slam Toolbox exhibits greater precision compared to GMapping and Hector SLAM. The robot's motion aligns more accurately with the virtual world when using Slam Toolbox, in contrast to the others. However, the choice between these SLAM solutions depends on your specific project requirements, familiarity with the tools, hardware capabilities, and the complexity of your robot's operating environment. Each of these options has its strengths and weaknesses, and the better choice will vary depending on the context of your application.

As for human detection, the experimental results returned were quite surprising as the YOLOv8 model did not have outstanding results compared to older models; however, we see it still has potential for development, so we still choose it. Furthermore, because our initial goal was to apply to mobile robots with small microprocessors, this trade-off between processing speed and accuracy was extremely reasonable.

After this article, our next direction is to research more deeply into other more advanced tasks such as tracking, determine people's location, movements and behaviors of using 3D point cloud, etc. It will have some challenges such as privacy, security, fairness, scalability, and interdisciplinary collaboration will be addressed to ensure ethical and impactful innovation. But we hope that we can further apply modern technology to contribute to improving human happiness in today's society.

References

[1] M. Steve, M. Tom, L. David, M. Alexey, F. Michael. “From the Desks of ROS Maintainers: A Survey of Modern & Capable Mobile Robotics Algorithms in the Robot Operating System 2”, Robotics and Autonomous Systems, vol. 168, 2023. (https://doi.org/10.1016/j.robot.2023.104493)

[2] Huang, Jiahao, Steffen Junginger, Hui Liu, and Kerstin Thurow, "Indoor Positioning Systems of Mobile Robots: A Review" Robotics 12, no. 2: 47, 2023. https://doi.org/10.3390/robotics12020047

[3] Shibata T, Wada K. Robot therapy: a new approach for mental healthcare of the elderly - a mini-review. Gerontology. 2011;57(4):378-86. doi: 10.1159/000319015. Epub 2010 Jul 15. PMID: 20639620.

[4] Saputra, M.R., Markham, A., & Trigoni, A. (2018). Visual SLAM and Structure from Motion in Dynamic Environments: A Survey. ACM Comput. Surv., 51, 37:1-37:36.

[5] Zhou, H., & Zhang, T. (2014). The combination of SfM and monocular SLAM. The 26th Chinese Control and Decision Conference (2014 CCDC), 5282-5286.

[6] Pyo, Y., Jo, H., Jeong, R. and Im, T., (2017). ROS Robot Programming. Bucheon: Rubi Peipeo, pp.313:359.

[7] B. Abhishek, S. Gautham, D. Varun Rufus Raj Samuel, K. Keshav, U. P. Vignesh and S. R. Nair, "ROS based stereo vision system for autonomous vehicle," 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, 2017, pp. 2269-2273.

[8] Oajsalee, S., Tantrairatn, S., & Khaengkarn, S. (2019). Study of ROS Based Localization and Mapping for Closed Area Survey. 2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR), 24-28.

[9] Filipenko, M., & Afanasyev, I. (2018). Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment. 2018 International Conference on Intelligent Systems (IS), 400-407.

[10] Zhu, Xingkui, et al. "TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios." arXiv (Cornell University), Cornell University, Aug. 2021, https://doi.org/10.48550/arxiv.2108.11539.

[11] Li, Chuyi, et al. "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications." arXiv (Cornell University), Cornell University, Sept. 2022, https://doi.org/10.48550/arxiv.2209.02976.

[12] Wang, Chien-Yao, et al. "YOLOv7: Trainable Bag-of-freebies Sets New State-of-the-art for Real-time Object Detectors." arXiv (Cornell University), Cornell University, July 2022, https://doi.org/10.48550/arxiv.2207.02696.

[13] Reis, Donald J., et al. "Real-Time Flying Object Detection With YOLOv8." arXiv (Cornell University), Cornell University, May 2023, https://doi.org/10.48550/arxiv.2305.09972.

[14] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey and K. Konolige, "The Office Marathon: Robust navigation in an indoor office environment," 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 2010, pp. 300-307, doi: 10.1109/ROBOT.2010.5509725.

[15] Macenski S, Tsai D, Feinberg M. Spatio-temporal voxel layer: A view on robot perception for the dynamic world. International Journal of Advanced Robotic Systems. 2020;17(2). doi:10.1177/1729881420910530

[16] R P Guan, B Ristic and L Wang, "KLD sampling with Gmapping proposal for Monte Carlo localization of mobile robots", Information Fusion, no. 49, pp. 79-88, 2019.

[17] Dwijotomo, A.; Abdul Rahman, M.A.; Mohammed Ariff, M.H.; Zamzuri, H.; Wan Azree, W.M.H. Cartographer SLAM Method for Optimization with an Adaptive Multi-Distance Scan Scheduler. Appl. Sci. 2020, 10, 347. https://doi.org/10.3390/app10010347

[18] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, andR. Vincent, "Efficient sparse pose adjustment for 2D mapping," IEEE/RSJ2010 Int. Conf. Intell. Robot. Syst. IROS 2010 - Conf. Proc., no. October,pp. 22–29, 2010.

[19] Macenski, Steve & Jambrecic, Ivona. (2021). SLAM Toolbox: SLAM for the dynamic world. Journal of Open Source Software. 6. 2783. 10.21105/joss.02783.

[20] Terven, Juan R., and Diana Cordova-Esparza. "A Comprehensive Review of YOLO: From YOLOv1 and Beyond." arXiv (Cornell University), Cornell University, Apr. 2023, https://doi.org/10.48550/arxiv.2304.00501.

[21] Murphy, K., Russell, S. (2001). Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In: Doucet, A., de Freitas, N., Gordon, N. (eds) Sequential Monte Carlo Methods in Practice. Statistics for Engineering and Information Science. Springer, New York, NY. https://doi.org/10.1007/978-1-4757-3437-9_24

[22] K. Murphy, "Bayesian map learning in dynamic environments," in Proc. Conf. Neural Inf. Process. Syst., Denver, CO, 1999, pp. 1015–1021.

[23] Trejos K, Rincón L, Bolaños M, Fallas J, Marín L. 2D SLAM Algorithms Characterization, Calibration, and Comparison Considering Pose Error, Map Accuracy as Well as CPU and Memory Usage. Sensors. 2022; 22(18):6903. https://doi.org/10.3390/s22186903

[24] W. Hess, D. Kohler, H. Rapp and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 2016, pp. 1271-1278, doi: 10.1109/ICRA.2016.7487258.

[25] Ma Yi-de, Liu Qing, and Qian Zhi-Bai. Automated image segmentation using improved pcnn model based on cross-entropy. In Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004., pages 743–746. IEEE, 2004.

[26] Zhang, Haoyang, et al. "VarifocalNet: An IoU-aware Dense Object Detector." arXiv (Cornell University), Cornell University, Aug. 2020, https://doi.org/10.48550/arxiv.2008.13367.

[27] Hossain, Sazzad, et al. "Dual Focal Loss to Address Class Imbalance in Semantic Segmentation." Neurocomputing, vol. 462, Elsevier BV, Oct. 2021, pp. 69–87. https://doi.org/10.1016/j.neucom.2021.07.055.

[28] Rezatofighi, Hamid, et al. "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression." arXiv (Cornell University), Cornell University, Feb. 2019, https://doi.org/10.48550/arxiv.1902.09630.

[29] Zheng, Zhaohui, et al. "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression." arXiv (Cornell University), Cornell University, Nov. 2019, https://doi.org/10.48550/arxiv.1911.08287.

[30] Bochkovskiy, Alexey, et al. "YOLOv4: Optimal Speed and Accuracy of Object Detection." arXiv (Cornell University), Cornell University, Apr. 2020, arxiv.org/pdf/2004.10934v1.

[31] Ding, Xiaohan, et al. "RepVGG: Making VGG-style ConvNets Great Again." arXiv (Cornell University), Cornell University, Jan. 2021, https://doi.org/10.48550/arxiv.2101.03697.

[32] Wang, Chien-Yao, Hong-Yuan Mark Liao, et al. "CSPNet: A New Backbone That Can Enhance Learning Capability of CNN." arXiv (Cornell University), Cornell University, Nov. 2019, https://doi.org/10.48550/arxiv.1911.11929.

[33] Loshchilov, Ilya, and Frank Hutter. "Decoupled Weight Decay Regularization." International Conference on Learning Representations, Sept. 2018, openreview.net/pdf?id=Bkg6RiCqY7.

[34] Reddi, Sashank J., et al. "On the Convergence of Adam and Beyond." arXiv (Cornell University), Cornell University, Feb. 2018, arxiv.org/pdf/1904.09237.

[35] Voxel. "GitHub - Voxel51/Fiftyone: The Open-source Tool for Building High-quality Datasets and Computer Vision Models." GitHub, github.com/voxel51/fiftyone.niversity), Cornell University, Apr. 2020, arxiv.org/pdf/2004.10934v1.

[36] Lin, Tsung-Yi, et al. "Microsoft COCO: Common Objects in Context." Lecture Notes in Computer Science, 2014, pp. 740–55. https://doi.org/10.1007/978-3-319-10602-1_48.

[37] R. Padilla, S. L. Netto and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Niteroi, Brazil, 2020, pp. 237-242, doi: 10.1109/IWSSIP48289.2020.9145130.

[38] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai and R. Vincent, "Efficient Sparse Pose Adjustment for 2D mapping," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 2010, pp. 22-29, doi: 10.1109/IROS.2010.5649043.

[39] Ehsanpour, M., et al., JRDB-Act: A Large-scale Dataset for Spatio-temporal Action, Social Group and Activity Detection. 2021, arXiv.