Vigilada Mineducación

FEASIBILITY OF USING MONOCULAR VISUAL SLAM WITH ROS FOR ROBOT NAVIGATION IN WAREHOUSES

Viabilidad del uso de Visual SLAM monocular con ROS para la navegación de robots en bodegas

OSWALDO JOSÉ PARADA CUADROS Y DAVINSON CASTAÑO CANO

Tesis de grado

Asesor

Davinson Castaño Cano

UNIVERSIDAD EAFIT

Escuela de Ingeniería

Maestria en Ingenieria

Medellín

2023

Feasibility of using Monocular Visual SLAM with ROS for robot navigation in warehouses

Oswaldo Parada[1] and Davinson Castaño-Cano[1]

***Abstract*— In recent years, the use of robots in industrial facilities has become widespread. Particularly, the logistics and inventory management industry has benefited from technological advances, however, the high costs of these technologies prevent many companies from adopting this degree of automation. In this research, we propose the feasibility of using monocular Visual SLAM as an alternative to reduce costs in the localization hardware used for robot navigation in warehouses. For that, we study open source implementations, especially those based on ORB-SLAM2, ROS and Ubuntu. Then, we propose a testing environment compose by a Turtle Bot 2 robot and a location that ensembles some of the characteristics of warehouses.**

Our analysis shows that it is possible to use monocular Visual SLAM in warehouses, however, we mention some elements in the software and hardware that should be fixed or improved in current implementations in order to robustly and safely operate robots in an indoor storage environment.

I. INTRODUCTION

Autonomous mobile robots (AMRs) have proven to be very useful in storage locations, especially for storing or picking up cargo without human intervention [1]. This robotic application constantly promotes itself as a way to save considerable amounts of money, improve times and efficiency in logistics activities. Furthermore, the use of robotics has reduced the number of fatalities, accidents and occupational diseases in warehouse workers.

However, as the use of AMRs clearly represents benefits, few companies can successfully implement this kind of technologies mostly because initial money investment is too high. An important factor in why these implementations are so expensive is closely related to the navigation technology used by these robots. The traditional techniques for autonomous navigation inside storage facilities is by reading fiducial markers [2] or RFID tags that are logically placed on the floor [3]. Both options work quite similar, each fiducial markers or RFID tag have a unique identifier that can be read by the robot to then query it in a database to get the exact coordinates where the tag is located inside the building and where is the next tag the robot should read to get close to its final destination. Additionally for fiducial markers, robots could even detect if they are misaligned with their planned path by checking the angle in which the camera reads the code [4].

Fiducial markers are not expensive as anyone could create them using online apps and placing them can be as easy as sticking them to the floor using transparent tape. Nonetheless, robots have to read the markers in movement, meaning the camera mounted in the robot chassis must be good enough to read codes at a certain speed. These types of cameras can highly increase the cost of navigating with fiducial markers.

In the case of RFIDs implementation, the big downside could be the installation of the tags in the warehouse floor. Both fiducial markers and RFIDs have an additional problem: if the warehouse layout changes, e.g. the shelves are moved around, the tags or fiducial markers must also be relocated and remapped.

For the above reasons, new alternatives are being implemented for robot navigation inside warehouses. In particular, alternatives involving images are being explored, since storage facilities have very favorable characteristics for this kind of technology. For example, warehouses usually have controlled artificial lighting, which allows changes in light intensity, shadows, and light color not to affect the images, as can happen in outdoor environments.

That's why we decided to explore Simultaneous Localization and Mapping (SLAM), a technique widely used in recent years in autonomous robot applications because it is a ready-to-go navigation method for any type of location. Nevertheless, traditional SLAM techniques require specialized sensors that tend to be expensive or difficult to access in some parts of the world. As a result, many efforts both in the academic and in the industry field have been devoted to the development of Visual SLAM (v-SLAM), a variation of SLAM that proposes the use of cameras as the main input source, which represents greater ease in terms of economic accessibility and availability [5].

II. METHODOLOGY

In this paper, we analyze the feasibility of using Visual SLAM with a monocular camera as an alternative method for robot navigation in warehouses. For that, we first define the components that represent our environment:

A. AMR

We use a TurtleBot 2 robot to carry out the experiment. Even if it is not a robot suitable for the task of working in a warehouse, it has all the component of a platform suitable for such facility. TurtleBot 2 has multiple hardware pieces that allows us sense the environment around the robot, however, due to the nature of this research, we are interested in the following elements:

1) Kinect: RGBD cameras produce RGB images as regular cameras but additionally it gives information about how far elements are from the sensor by using infrared projectors.

*This work was not supported by any organization

[1]EAFIT University, Carrera 49 N° 7 Sur-50, Medellín, Colombia oparada@eafit.edu.co

Fig. 1: On the left: A visual representation of the Learning Factory laboratory from above. The gray areas are movable objects. On the right: One of the hallways in the Learning Factory Laboratory.

In the context of this paper, the results obtained with this camera will be considered as the ground truth since its efficiency in RGBD SLAM has been demonstrated [6].

In order to use Kinect as ground truth for the experiments, we use the turtlebot_app ROS library [7], a set of packages with the necessary tools for mapping and localization using an RGBD camera. Specifically, we can create maps using Gmapping and we can navigate the location using AMCL.

2) Odometer: The TurtleBot 2 we use has an odometer with around 25,000 ticks per revolution, which allow us to get accurate information on the robot's odometry. This kind of data is important in the navigation of the robot.

Other hardware pieces such as bumpers and gyroscopes are important in the operation and safety of the robot but are not considered key pieces in the results of this investigation.

B. Camera

Choosing the camera is not an easy task since it must be good enough to produce quality images at high frame rates, but it must also be easily accessible and cost-effective. The camera we use in this project is a Logitech webcam C270, which has 720p resolution at a rate of 30 fps. This camera has a cost of around 25 USD. It should be noted that this camera is monocular.

C. Location

Access to real and operational warehouses can be difficult as these locations are generally very busy, but overall, warehouses are not a place for unauthorized or untrained personnel due to all the latent risks that can occur. Fortunately, EAFIT University has multiple rooms with characteristics similar to those of a warehouse. In our case, we use The Learning Factory laboratory, a space that emulates an assembly line of a production factory. This laboratory does not fully represent a warehouse but it has very similar attributes such as floor-to-ceiling shelves, controlled lighting, well-defined corridors and bare floors. Figure 1 shows the layout of the laboratory and an example of the elements that are found inside.

D. Software

We decide to use Ubuntu 18.04.6 LTS as the operating system where other software will run. Ubuntu has been widely adopted by robotics researchers since several versions of ROS have been optimized for this OS.

We use ROS (Robot Operating System) which is a software known for its set of libraries and tools that makes the programming of robots easier. By using ROS, we don't have to worry about robot navigation logic, motor control, odometry calculation, sensor data capturing, etc. because there are out-of-the-shelf libraries we can plug and play in our routines.

The main nodes that make up the ROS context in this research are:

- Navigation Stack: This node enables the control of the robot's mobile base by sending velocity commands based on the information from odometers, sensors, map and destination [8].
- usb_cam: This library allows us to read any USB camera and stream the images as a ROS topic to which other topics can subscribe [9].
- camera_calibration: We use this library to calibrate the camera using a checkerboard calibration target. The library allows us to characterize the intrinsic distortions of any camera that we can use later to make sure our visual SLAM algorithm receives the images more faithful to reality [10].
- RViz: It is worth mentioning that we use RViz, which is a 3D visualization tool for ROS that allows us to see and control the elements generated by the execution of a robot application [11].

E. Computer

We use a MSI Laptop GL62M model with Intel core I7 7th generation, 16GB RAM and 256GB SSD. This computer is used both as a development environment and as the compute platform of the robot.

F. Monocular visual SLAM Algorithm

We concentrate our efforts in the selection of which Visual SLAM algorithm we should use to assert the hypothesis of this paper. The reason is because Visual SLAM is a technique that has been widely studied in recent years which has generated a wide variety of methods, each one with different characteristics [12]. Fortunately, our use case defines what are the criteria that we must take into account when choosing an algorithm:

1) Algorithm Type: Several v-SLAM algorithms can be classified as direct, feature-based, or hybrid methods. Direct methods are those that use object detection techniques on the images received from the camera, such as Dense tracking and mapping (DTAM)[13] and Direct Sparse Odometry (DSO)[14]. Another type is feature-based, that obtains descriptive points from each frame captured by the image and save them in a point cloud that the same algorithm can use to estimate the pose of the camera by comparing the live stream features of the camera with the stored features. A major drawback of feature-based algorithms is that it requires the environment to have distinctive characteristics in each frame for the algorithm to work correctly. Hallways or walls painted

the same color are examples where a feature-based algorithm doesn't work well. In this work, we focus on a very popular featured-based algorithm known as ORB-SLAM2 [15].

Additionally, there is a subgroup of algorithms that aims to improve depth calculation and object detection by integrating machine learning techniques [16]. In this study, we use a depth calculation method with a deep learning technique known as Fourier Convolutional Neural Networks (FCNN).

2) Loop Closure: Loop closure allows the v-SLAM algorithm to identify images that have already been detected, thus enabling the ability to correct for drift that may have occurred during mapping. This characteristic is very important in the navigation of indoor spaces. It is also an essential feature in monocular visual SLAM as the lack of an intrinsic depth parameter causes maps to lose proportions.

3) Scale Estimation: Monocular visual SLAM algorithms have the downside of not calculating an accurate scale since it only has the input of a single camera that does not give additional information about depth. However, there are a couple of strategies that solved this problem [17]. One option is to know the real size of an object in the location and place the camera at a fixed distance from the object, get the relative size of the object in the image and then calculate the respective scale by comparing real size with relative size of the object. Another option is to use information from another sensor to calculate the scale, the most used option that suits our conditions is to use the robot's odometry, where we can capture a frame at a specific location spot, move the robot a predefined distance perpendicular to the frame and capture a new image, this way we can calculate the difference between relative object size in frame 1 and frame 2.

4) ROS Compatibility: Most Visual SLAM algorithms were not specifically developed to run on ROS, many began as standalone scripts that run on a computer without the need of being run directly on a robotic platform. Fortunately, the open source community and researchers have adapted some algorithms to be run as a topic in ROS allowing the integration of the entire suite of robot control libraries with v-SLAM output.

G. Viability criteria

Now we describe what criteria we take into account to study the viability of navigation using monocular visual SLAM in warehouses, each of these items have their own evaluation criteria:

- Software: Here we assert if available software is enough to deploy a fully functional monocular visual SLAM application. The criteria questions are:
  - Is there a visual SLAM solution that we can use as a base for warehouse worker robot applications?
  - Is it possible to adapt an existing visual SLAM solution to our use case?
  - What are the changes that anyone must make to adapt the solution?
- Scale estimation: We evaluate if it is possible to calculate a correct scale factor using the components mentioned in Section II.
- Localization and Mapping: In this criteria we evaluate if localization and mapping features are good enough for the use case. The questions are:
  - Do the generated maps capture the real characteristics of the location?
  - How much noise is introduced to the map when using feature-based visual SLAM algorithms?
  - Is it possible to correctly estimate the pose of the robot with monocular visual SLAM?
  - What are the locations in which localization with v-SLAM does not give good results?
- Navigation: We want to evaluate if it is possible to navigate the robot through a warehouse by answering the following questions:
  - Is it possible to navigate using a monocular visual SLAM?
  - What are the locations in which navigation with v-SLAM does not give good results?
  - How accurate can the navigation of a robot using a monocular visual SLAM be?

H. Experiment description

We simulate the actions that a robot would carry out in a warehouse. In particular, we focus on navigation, since actions such as lifting or depositing objects, reading barcodes, among other typical actions in the warehousing industry will not be in the scope of this project. From there, we propose two experiments.

The first experiment consists of creating the map of the laboratory using both the Kinect and the monocular camera and then use RViz to measure the size of 3 objects that can be found in the scene: a table, a column and a shelf. After this, we want to calculate the error percentage between the measurement obtained from the mapping and the real size of the object with the objective of estimate the quality of the scale estimation by the algorithms.

The second experiment consists of moving the TurtleBot from a start point (S) to three different points (A, B and C) within the described location in section II-C and as can be seen in Figure 2. Each destination point is a trajectory described below:

1) Straight Line: The robot must move from point S to point A following a straight line of 6 meters long.
2) One Turn: The robot must move from point S to point B, which requires a turn in the trajectory .
3) Two Turns: The robot must move from point S to point C passing from point A and B and describing a trajectory with two turns.

In each iteration, we log the following data:

- The robot managed to complete the test. This is a true/false answer
- The difference in centimeters between the destination and the robot.

All trajectories mentioned above will be executed using the Kinect camera from the TurtleBot, and its results will be

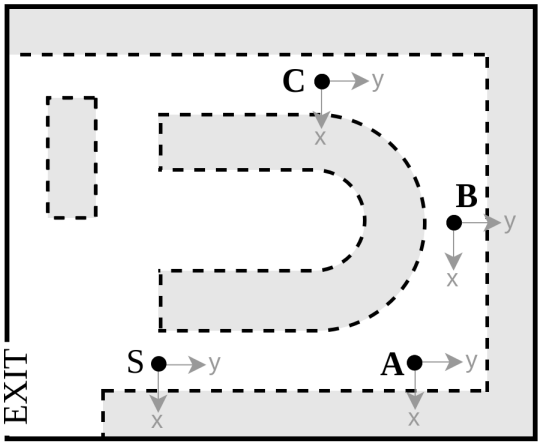

Fig. 2: Location of the start point (S) and the destination points (A, B and C) in the Learning Factory laboratory. The coordinate system references are shown in darker gray.

recorded as the ground truth to compare with the selected monocular visual SLAM solution.

An overall representation of the components mentioned in this section are shown in Figure 3. In this figure, we illustrate the relationship between each software and hardware element and how their outputs serve as input to other components, which is very useful when understanding the next section.

III. EXPERIMENT AND DISCUSSION

We discuss the result of the experiments based on the viability criteria explained in Section II-G:

A. Software

We tested multiple open source implementations of monocular simultaneous localization and mapping adapted to ROS, all of them are open sourced and available in Github repositories. Initially, we tried the machine learning approach based on an implementation of FCNN (Fourier Convolutional Neural Networks) for the calculation of depth in monocular images [18]. It can be run on an NVidia Jetson TX2 which means that robots do not need large computer power to run the routines. Although the authors mentioned that ROS kinetic or a higher version can be used, at the beginning we tried to run it with ROS Melodic but there were several library version compatibility errors so we had to use ROS Kinetic. After a stable version was running in the robot, we carried out preliminary tests where we came to the conclusion that this implementation does not work for us because it only covers the use case of mapping and not localization as such.

Then, for the second iteration, we used a Visual SLAM project based on ORB-SLAM2 [19]. We highlight 2 characteristics of this application: The first one is the ability to save the point cloud captured with the algorithm, which enables the re-usability of the same point cloud to continue mapping later or simply use it for localization. The second characteristic is the use of three types of cameras: RGBD, Stereo and monocular, in addition to the use of any camera if the intrinsic properties of the hardware are known. Here we used ROS Melodic installed on Ubuntu 18 as the repository author recommended. The installation required fixing some library compatibility issues in order to make it work. This was one of the algorithms with which we carried out the most exploration and tests. We even managed to do end to end tests where we adapted the visual SLAM output with the ROS navigation stack to navigate the TurtleBot through the map. However, we found a big limitation of monocular v-SLAM approaches not being addressed in the implementation: the estimation of the scale. ORB-SLAM2, in the case of monocular, makes an estimate of the average depth of the initial scene and uses that factor for the rest of the execution, therefore, in new executions the scale varies with respect to the previous ones and causes a drift in the map scale. As a result, the robot's navigation was inaccurate and even on several occasions the robot could not move because it detected obstacles that did not exist.

TABLE I: Width of some objects in the location. The measurements were taken from the maps captured by the Kinect and the monocular ORB-SLAM2.

Object	Measure [cm]			Error [%]	
	Real	Kinect	Monocular ORB-SLAM2	Kinect	Monocular ORB-SLAM2
Table	102	103,5	104	1,9	2,5
Column	51	51,6	52	1,5	2,5
Shelf	94	94,3	95,2	0,3	1,2

Finally, we used our own modified version [20] of an implementation of ORB-SLAM2 with the scale estimation feature based on the robot's odometry [21]. Some of the new code we added to our implementation includes Turtlebot setup, Logitech camera parameters for ORB-SLAM2 and the instructions on how to install the required software for the experiments. Figure 4 is an example of an image captured from the Learning Factory laboratory and shows how ORB-SLAM2 identifies and marks the unique features of the scene. These features will become part of a point cloud that works as a map in the localization and navigation step.

B. Scale estimation

Results for the object measurement experiment are shown in Table I. We can see the percentage error for the Kinect measurements are less that 2% and for monocular ORB-SLAM2 is less than 3%. Even though there is a difference from the actual measurements, ORB-SLAM2 based map is scaled roughly to the actual scale, which gives us an idea of how good scale estimation approaches can be for monocular visual SLAM.

C. Localization and Mapping

In this particular case, localization with ORB-SLAM2 is a robust attribute because it works based on features directly extracted from the environment. Distinctly, our location is full of unique or distinctive elements that improve localization performance. However, localization is not without

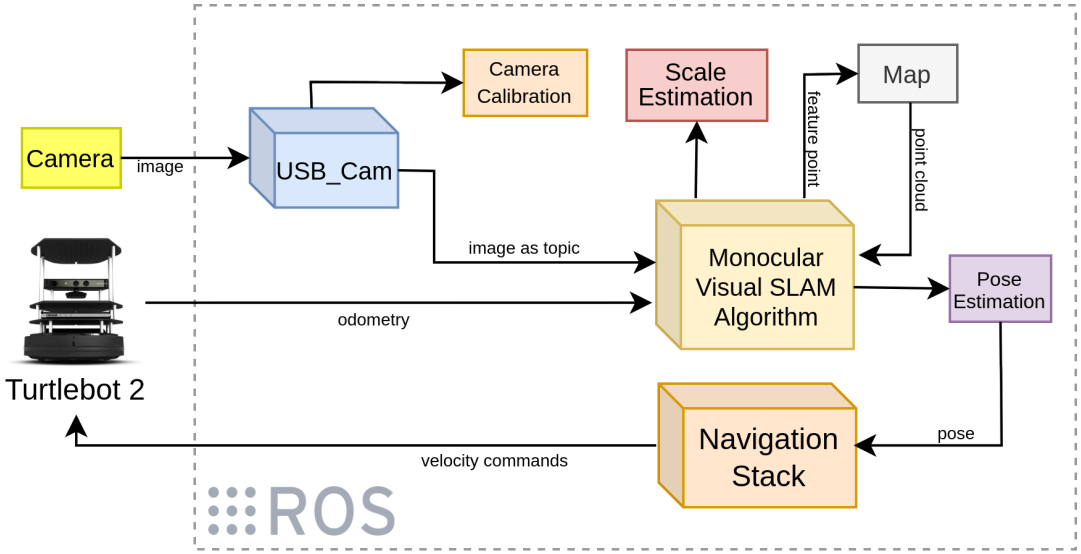

Fig. 3: Overall framework of monocular Visual SLAM with ORB-SLAM2, ROS and TurtleBot 2

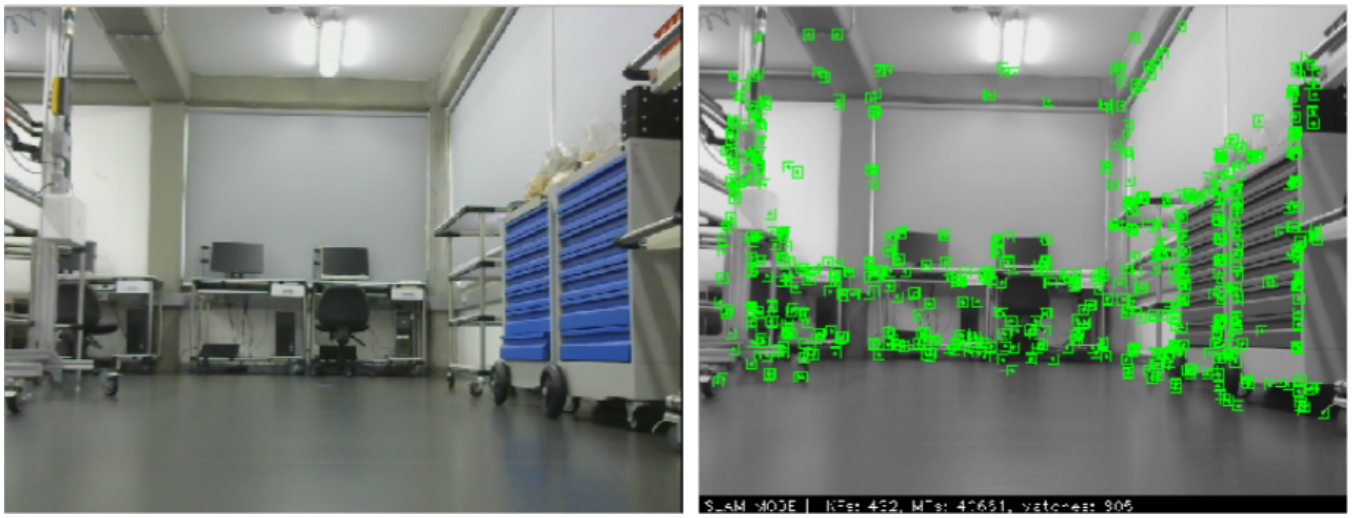

Fig. 4: On the left: raw image input from the camera. On the right: Same image after being processed by ORB-SLAM2. The points in the image are the features captured by the algorithm.

errors and can fail when the environment has changed a lot compared to previous maps.

Figure 5 shows the resulting map after mapping the entire location using the Kinect camera. Some characteristics of the location like the U shaped structure in the center or the exit door in the bottom left corner can be identified. The walls can be easily identifiable as well, however, elements such as storage racks or metal trusses are not captured well because the algorithm only uses points detected at a certain height. This means less detailed maps but at the same time maps with less noise.

Furthermore, Figure 6 shows the point cloud from above resulting of mapping the laboratory using the monocular camera with ORB-SLAM2. This type of map has better detailed object shapes than the map created with the Kinect camera, but it is also necessary to highlight that there is much more noise because the algorithm captures features from the entire scene, including the floor and ceiling. Note also, that there are points that are outside the limits of the laboratory, this is caused by the same algorithm when misidentifying a feature.

D. Navigation

The results of the second experiment can be seen in table II. In there, we can see that there is a considerable distance between the robot and point A for both methods. However, for the one turn trajectory, the distance when the Kinect was used is very small compared to the ORB-SLAM2 method. For the two turns trajectory, Kinect run still has a small distance to the destination point but on the other hand, this test could not be completed with ORB-SLAM2 because the algorithm detected obstacles that did not exist.

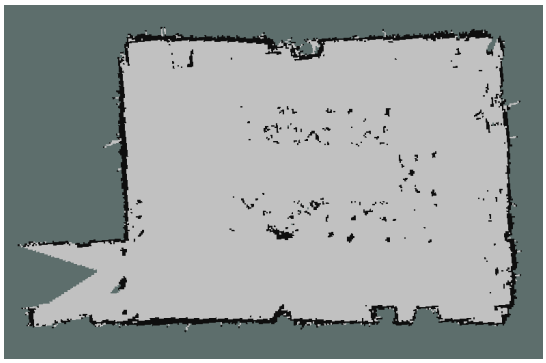

Fig. 5: View from above of the map resulting from the mapping of the laboratory using the kinect camera.

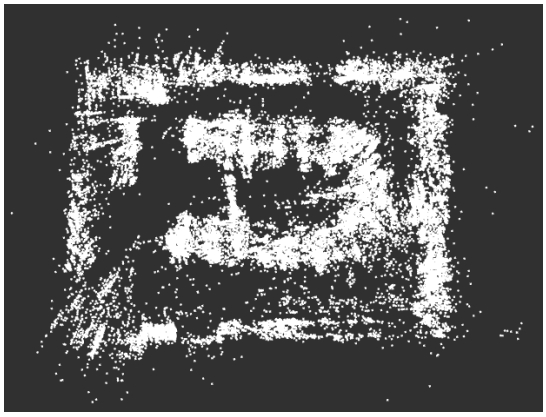

Fig. 6: View from above of the point cloud resulting from the mapping of the laboratory using ORB-SLAM2.

One explanation for this type of error is the number of additional points that the map shown in Figure 6 has.

IV. CONCLUSIONS

In this paper, we presented a feasibility study for using monocular visual SLAM as an alternative to robot localization and mapping in warehouses. Our experiments showed that it is possible to use monocular cameras as main sensors for SLAM techniques; however, there are still several software challenges to overcome in order to have a robust and safe implementation in warehouses. Furthermore, we conclude that the use of robot odometry can be useful to adjust the map scale when additional information about the depth of the scene is not available.

As future work, we want to study how the integration of additional sensors to the robot can help improve the shortcomings of monocular visual SLAM algorithms and add new safety elements when operating in warehouses where there are other robots and/or humans.

TABLE II: Distance between expected destination point and robot final position after navigating from the starting point to each one of the trajectories.

Method	Trajectory	Delta [cm]		Distance [cm]
		x	y	
Kinect	Straight Line	10	38	39,29
	One Turn	2,5	-3	3,91
	Two Turns	6	0	6
Monocular ORB-SLAM2	Straight Line	43	-26,5	50,51
	One Turn	-28	-15	31,76
	Two Turns	INC	INC	INC

REFERENCES

[1] A. Kattepur, A. Mukherjee, and P. Balamuralidhar, "Verification and timing analysis of industry 4.0 warehouse automation workflows," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2018, pp. 1297–1304.

[2] M. Kalaitzakis, B. Cain, S. Carroll, A. Ambrosi, C. Whitehead, and N. Vitzilaios, "Fiducial markers for pose estimation," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 4, p. 71, Mar 2021. [Online]. Available: https://doi.org/10.1007/s10846-020-01307-9

[3] S. Park and S. Hashimoto, "Autonomous mobile robot navigation using passive rfid in indoor environment," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 7, pp. 2366–2373, 2009.

[4] L. Ross and D. K. Bradshaw, "Fiducial marker navigation for mobile robots," *availabel at, Nov*, 2012.

[5] J. K. Makhubela, T. Zuva, and O. Y. Agunbiade, "A review on vision simultaneous localization and mapping (vslam)," in *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, 2018, pp. 1–5.

[6] J. Motsch, S. Benammar, and Y. Bergeon, "Interior mapping of a building: A real-life experiment with microsoft kinect for windows v1 and rgbd-slam," in *2017 International Conference on Military Technologies (ICMT)*, 2017, pp. 728–732.

[7] Turtlebot. (2017) Turtlebot_apps [Source Code]. https://github.com/turtlebot/turtlebot_apps.

[8] Ros-planning. (2022) navigation [Source Code]. https://github.com/ros-planning/navigation.

[9] Ros-drivers. (2022) usb_cam [Source Code]. https://github.com/ros-drivers/usb_cam.

[10] Ros-perception. (2022) image_pipeline [Source Code]. https://github.com/ros-perception/image_pipeline.

[11] Ros-visualization. (2022) rviz [Source Code]. https://github.com/ros-visualization/rviz.

[12] A. Macario Barros, Y. Moline, F. Carrel, M. Michel, and G. Corre, "A comprehensive survey of visual slam algorithms," *Robotics*, vol. 11, 02 2022.

[13] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 International Conference on Computer Vision*, 2011, pp. 2320–2327.

[14] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[15] M. J. M. M. Mur-Artal, Raúl and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[16] R. de Queiroz Mendes, E. G. Ribeiro, N. dos Santos Rosa, and V. Grassi, "On deep learning techniques to boost monocular depth estimation for autonomous navigation," *Robotics and Autonomous Systems*, vol. 136, p. 103701, Feb 2021. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2020.103701

[17] D. Zhou, Y. Dai, and H. Li, "Ground-plane-based absolute scale estimation for monocular visual odometry," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 791–802, 2020.

[18] A. Bokovoy, K. Muravyev, and K. Yakovlev, "Real-time vision-based depth reconstruction with nvidia jetson," 2019. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85074429057doi=10.1109%2fECMR.2019.8870936&partnerID=40&md5=b87bcba0803147012ee1062f867cc4ef

[19] AppliedAI. (2021) ORB-SLAM2 [Source Code]. https://github.com/appliedAI-Initiative/orb_slam_2_ros.

[20] ojpc. (2022) monocular-orb-slam-2-with-scale-correction [Source Code]. https://github.com/ojpc/monocular-orb-slam-2-with-scale-correction.

[21] Karwowski. (2020) ORB-SLAM2_ROS [Source Code]. https://github.com/rayvburn/ORB-SLAM2_ROS.git.