Intelligent Service Robotics (2023) 16:177–193
https://doi.org/10.1007/s11370-023-00457-7

ORIGINAL RESEARCH PAPER

A ROS-based distributed multi-robot localization and orientation strategy for heterogeneous robots

Abdussalam A. Alajami[1] · **Nil Palau**[1] · **Sergio Lopez-Soriano**[1] · **Rafael Pous**[1]

Received: 13 July 2022 / Accepted: 18 January 2023 / Published online: 23 February 2023

Abstract
The problem of estimating and tracking the location and orientation of a mobile robot by another in heterogeneous distributed multi-robots is studied in this paper. We propose a distributed multi-robot localization strategy (DMLS) that is Robotic Operating System (ROS) based. It consists of an algorithm that fuses data of diverse sensors from 2 heterogeneous robots that are not connected within their transform trees to localize and measure the relative position and orientation.
The method exploits the robust detection of the Convolutional Neural Networks (CNN) and the accurate relative position measurements from the local costmap. The algorithm is composed of two parts: The localization part and the relative orientation measurement part. Localization is done by optimization and alignment calibration of the CNN output with the costmap in an individual robot. The relative orientation measurement is done by a collaborative multi-robot fusing of diverse sensor data to align and synchronize the transform frames of both robots in their costmaps. To illustrate the performance of this strategy, the proposed method is compared with a conventional object localization and orientation measuring method that uses computer vision and QR codes. The results show that this proposed method is robust and accurate while maintaining a degree of simplicity and efficiency in costs. The paper also presents various application experiments in laboratory and simulation environments. By using the proposed method, distributed multi-robots collaborate to achieve collective intelligence from individuals, which increases team performance.

Keywords Multi-robots · Localization · Collaboration · Exploration

1 Introduction

Localization is one of the main requirements for the autonomy of a mobile robot [1]. In order to navigate autonomously in their workspace, mobile robots must be able to localize themselves, indoors and outdoors. To successfully perform the tasks required of them, mobile robots need to know their exact position and orientation (pose). There have been numerous approaches to the localization problem for a single robot utilizing different types of sensors and techniques. In [2], authors use a technique that combines position estimation from odometry with observations of the environment from a mobile camera for achieving accurate self-localization. In [3], authors present a Bayesian estimation and the Kalman filter for object tracking. In [4], authors develop two algorithms to register a range scan to a previous scan in order to compute relative robot positions in an unknown environment.

The development of multiple robot systems that solve complex and dynamic problems in parallel is one of the key issues in robotics research. The multiple robot system, in comparison with the single robot system, has the advantage of collecting and integrating multiple sensor data from different robots for different purposes and applications. For this reason, many robotic applications require that robots are able to collaborate with each other within a team in order to perform a certain task [5]. An interesting application for multiple

Nil Palau and Sergio Lopez-Soriano have been authors contributed equally to this work.

✉ Abdussalam A. Alajami
Abdussalam.alajami@upf.edu

Nil Palau
nil.palau01@estudiant.upf.edu

Sergio Lopez-Soriano
sergio.lopez@upf.edu

Rafael Pous
rafael.pous@upf.edu

[1] Department of Communications and Information Technology, Universitat Pompeu Fabra, Roc Boronat, 138, 08018 Barcelona, Spain

robot' collaboration in [6], where authors aim to maintain an accurate and close to real time inventory of items using Radio Frequency Identification technology (RFID), which is considered crucial for an efficient Supply Chain Management (SCM). They first define the problem of stock counting and then a solution based on a multi-robot system is proposed.

However, in order for robots to increase their collaboration performance, they will need to be able to localize each other in their own maps, which opens up a wide degree of new applications such as map sharing for exploratory robots in [7], or map building and cooperative collaboration in [8], and other [9]. However, this is not a straightforward process if they do not share a common map or no prior information is given to the robots.

The advantages that derive from the exchange of information among the members of a team are more crucial in the case of heterogeneous team robots. A team of robots that is composed of different types or platforms carrying different proprioceptive and exteroceptive sensors have different capabilities for sensing the environment and self-localization. The exploitation of the heterogeneity feature in distributed multi-robots allows researchers to find different strategies for cooperative multi-robot localization.

This paper proposes a Robotic operating system (ROS)-based multi-robot localization and relative orientation measurement strategy for distributed heterogeneous robots. The proposed strategy or method helps increase the efficiency of the collaboration between heterogeneous multi-robots in a team by addressing the problem of accurately detecting and localizing multi-robots in unknown and un-explored environments autonomously and through cooperation. The proposed method exploits the capability of the heterogeneity feature that exists in a group of robots. It fuses data of diverse sensors from 2 heterogeneous robots that are not connected within their transform trees to mutually localize and measure their relative orientation. The method exploits the robust detection and tracking of Convolutional Neural Networks (CNNs) and the accurate relative distance measurements from a local costmap. The exact localization and orientation measurement are done when two members of a robot team are in close range from each other, this is done by utilizing a stereo camera that exists on only one member of the team, and the 2D local costmaps from both robots that are constructed using precise 2D-lidars.

2 Similar work

Previous similar work has been done considering collaborative strategies for multi-robot localization. A number of authors have considered pragmatic multi-robot map-making. Some approaches use beacon-type sensors. For example, in article [10], authors propose an algorithm for model-based localization that relies on the concept of a geometric beacon. The algorithm is based on an extended Kalman filter that utilizes matches between observed geometric beacons and a priory map of beacon locations. The fact that a prior reference map was given and shared between the robots, meant that the robot's transform trees were somehow connected. This made it relatively straightforward to transform observations from a given position to the frame of reference of the other observers. The algorithm exploited structural relationships in the data.

In other work [11], Rekleitis, Dudek, and Milios have demonstrated the utility of introducing a second robot to aid in the tracking of the exploratory robot's position and they introduced the concept of cooperative localization. Their approach is based on using pairs of robots that observe each other's behavior, acting in concert to reduce odometry errors. Their approach improves the quality of the map by reducing the inaccuracies that occur over time from dead reckoning errors. However, prior knowledge of their initial location was also known, which reduced the autonomy of the system. Vision-based systems have been widely used in robotic perception sensing. They are also useful for distinguishing an individual robot for handling the case of more than two robots. In the past few years, several authors have considered localizing team members of a group of robots using each other. They used various types of sensors for this objective including vision-based sensors. An example in [12] proposes an algorithm that allows robots to identify and localize each other. The method requires all robots to be equipped with omnidirectional mono cameras. Jennings et al. in [13] used stereo vision to build a grid map and localize robots by features in the grid maps such as corners. Their key idea is to find corners in a grid map and compare these corners with a-priory landmarks at known positions. They needed an exact reference of the map for their method to work, which makes it not useful for exploration-based robots. Fox et al. proposed a multi-robot Monte Carlo Localization (MCL) approach in [14], to localization with improved accuracy of MCL. Their research shows accurate localization results and can be used for multi-robot localization. However, it requires the transmission of a relatively large amount of information. Pereira et al. in [15] proposed an approach to localize and track objects exploiting statistical operators and a simple graph searching algorithm. In [16], distributed sensing was developed based on Kalman filtering to improve the target localization. However, these methods are only designed to recognize and localize a specific entity or target. In a multi-robot scenario where robot members collaborate, it is preferable and logical for each robot to localize and track all of its teammates, also important features in the environment. Researchers have also used CNNs for high-performance features and robot identification and tracking.

In [17], authors using CNNs present a robust multi-robot convoying approach that relies on visual detection of the leading agent, thus enabling target following in unstructured 3-D environments. However, although this method detects the other robot with high performance and measures the relative motions of that robot, it does not precisely localize the other robot nor measure its relative orientation. The distributed multi-robot localization strategy (DMLS) introduced in this paper, is designed to operate with the sensors or hardware that normally exists in an autonomous mobile robot that is able to construct a local costmap of his surrounding environment. However, within a team of two robots, one of the robots would require to be equipped with a camera. The camera can be the same proximity sensor (e.g., RGBD camera) or the self-localization tracking camera that contributes in constructing a local costmap in the robot. The DMLS strategy exploits the strengths of the diverse sensors that exist in the heterogeneous team robots. It utilizes the robust detection and tracking in CNNs and the accurate proximity measurements extracted from a 2D local costmap that is constructed using a 2D-lidar for the accurate localization and relative orientation estimations between the robots.

The detailed description of the heterogeneous robots utilized in this paper is introduced in Sect. 3. Section 4 presents the framework of the proposed strategy. In Sect. 5, the proposed localization method will be compared with a vision-based conventional QR-pose estimation method. The QR-pose estimation method was used due to its low computation and flexibility to operate on limited machines such as Unmanned Aerial Vehicles (UAVs) and simple distributed Unmanned Ground Vehicles (UGVs). It also operates in decentralized systems where both robots do not share prior information or a map. Finally, the fact that makes it able to estimate position and relative orientation at the same time makes it a good comparison reference to be used. Some application scenarios in Sect. 6 illustrate the contribution of the proposed method using a laboratory environment and a ROS-based simulation platform (Gazebo). Finally, Sects. 7 and 8 present the overall conclusions and future work.

3 System overview

For the experiments executed to implement and test the DMLS method introduced in this paper, two independent heterogeneous robots are used, called *Detector* and *Pawn*. These robots are designed to be able to cooperate on a shared task. The robots are able to communicate using a common network.

For the experiments run in the laboratory, the *Detector* robot will be a small sized UAV, the *Detector*'s hardware block diagram is shown in Fig. 1. The UAV was designed to

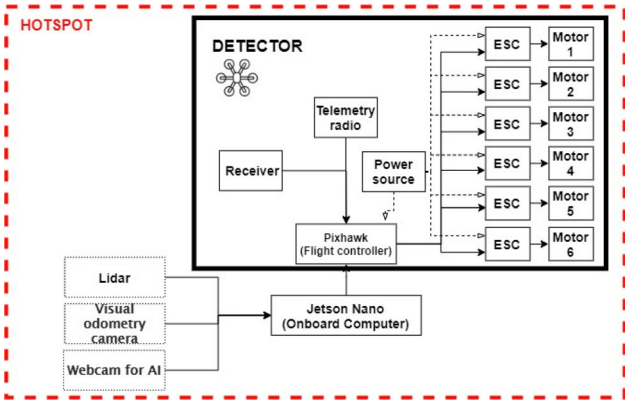


Fig. 1 Hardware block diagram of the *Detector*

be able to execute stable navigation in indoor spaces; therefore, a 6-motor UAV (hexacopter) frame was chosen. The frame design layout enables a stable flight in indoor spaces for a task-sufficient amount of time. An efficient open-source autopilot (Pixhawk 2.4.8) is used. A compatible companion computer (Jetson nano) is used as the main processing unit for the UAV. The UAV also contains 6 brush-less motors (750 kv), 6 propellers (16"× 4" size), and 6 electronic speed controllers (ESCs 40Amp of current discharge capacity). The ESCs main objective is to translate the signal received from the autopilot to energy from the energy source and supply it to the motors. The energy source used on the UAV is a lithium polymer battery (LIPO, 16v, 6cell), which capacity is 6000 mAh, operating voltage is 16.6v, and can supply a high current discharge rate.

Since the UAV was intended to operate in indoor spaces (GPS-denied spaces), a Visual Simultaneous Localization and Mapping (VI-SLAM)-based camera was added to the system to supply the UAV with self-localization messages. The VI-SLAM camera used, has a built in diverse suite of sensors which all feed into a VI-SLAM pipeline, which fuses them into a 6 DOF estimation of position and velocity of the camera relative to the environment at 200 Hz, therefore, precisely tracking and self-localize the UAV. Special adaptation was made to integrate these messages into the autopilot, resulting in an indoor guidance system for the UAV. For the Experiments, the *Pawn* robot is a UGV and its hardware block diagram is shown in Fig. 2.

The pre-designed and patent UGV in [6] is used. The UGV is designed to autonomously navigate within the given path in the intended map layout. In order for the robots to be able to run the proposed localization method and successfully localize each other, the *Detector* must have a camera mounted for detection. This is not a requirement for the *Pawn*. However, sufficient sensors that enable the construction of a local costmap will be necessary for both the *Detector* and the *Pawn*. For the laboratory Experiments, a UAV in [18],

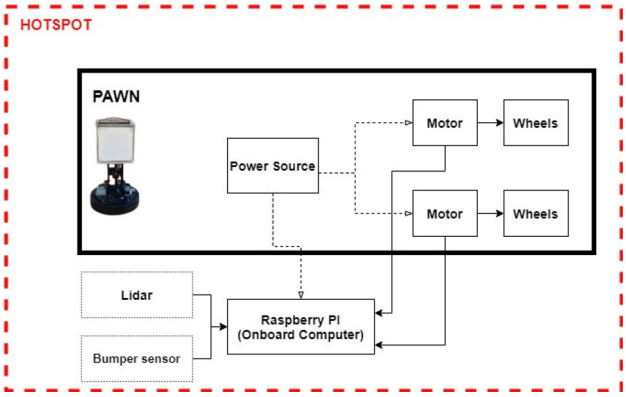


Fig. 2 Hardware block diagram of the *Pawn*

has been chosen to be the *Detector* for the manifest advantage of utilizing 3D space. This increases the possibility of detection while hovering or flying. However, the UAV must land, or be in the same 2D plane as the *Pawn* after detection. This step is necessary for completing the sequential process of the proposed localization method, DMLS.

4 DMLS framework

The proposed localization method consists of four parts.

4.1 First part: detecting and reducing the region of search (RoS) using CNN

Throughout a mission, the *Detector* would first detect the *Pawn* as soon as it is in the detection range of its camera and in line of sight (LoS). This is done by using a high-performance, cost-effective, and low-computation CNN model called "MobileNetV2" [19]. The MobileNetV2 model developed by Google, is designed to dramatically reduce the complexity cost, and model size of the network. It is specially optimized to operate on mobile devices. The architecture delivers high-accuracy results while keeping the parameters and mathematical operations as low as possible. This makes it adequate for resource-limited mobile robots. The objective of this CNN is to identify and classify the class/object of the detected robot from a list of pre-trained classes/objects. For our experiments, we trained the model with a dataset that is composed of around 2000 high-resolution images per class/object that is required to be detected. The images for the dataset are taken from different angles, distances, and illumination environments to assure the accurate detection of the class/object at any condition. Transfer learning was used on pre-trained existing models supported by the API. A total of 91 epochs (around 20h of training time), were used to train the model with 6 classes/objects which include the *Pawn* and some important features of the laboratory environment. A boundary box will be plotted on the object with the most likelihood (MLH) estimation to be the *Pawn*. The pixel positions of the boundary box in the x-axis and y-axis of the entire pixel space *PS* of the image frame are provided by the CNN. The minimum and maximum pixel position values on the horizontal axis (*PS_H*) of the boundary box will be recorded as $x\text{pix}_{\text{min}}$ and $x\text{pix}_{\text{max}}$ as only the x-axis values will be used in the calculations of the RoS.

4.2 Second part: camera frame and costmaps calibration and locating

The constructed local costmap of the robot is done by a ROS [20] package, called Move_base [21]. This is achieved using proximity sensors mounted on the robot. Move_base is a simple algorithm that constructs a matrix C, which represents a local costmap. The element c_{ij} represents a cost function in every cell of the costmap, or so called cell-cost [22]. This value represents the possibility of having an obstacle within that cell.

However, due to the fact that the size of the local costmap of a robot differs from one robot to another depending on the sensors type, detection range, and local costmap parameters configurations, this implies that there will be a misalignment of the size of the local costmap (width and height) with the horizontal field of view (HFOV) of a camera frame. In other words, the local costmap might include more obstacles in the environment than what the camera frame is able to visualize as shown in Fig. 3. As a result, a calibration/alignment step is needed to measure the effective local costmap that relates to the camera HFOV, which is represented as the pink highlighted area in Fig. 3. Most local costmaps are usually bigger in size than the HFOV of a camera frame (if not, it can be set so by optimizing the local costmap size parameter). The effective local costmap will be a submatrix of the overall matrix that represents the entire local costmap. The elements C' and C'_{ij} represent the submatrix and the cost function in every cell of the effective costmap.

The calibration step consists of the measurement of the boundaries of this submatrix. By placing the robot in an empty obstacle-free area of space, and at a set distance $0 < d < costmap - height/2$ from the working distance [23] in front of the robot and camera, an object is introduced so it can be visible at the edge of the left side of the image frame. From the local costmap a cell with a high-cost value $c_{ij} = 100$ will be created, the column position of this cell will then be recorded j'_{min} representing the left boundary or the starting column position of the submatrix. The same process will be repeated but this time introducing an object from the right side of the image frame to measure j'_{max}, which represents the column position of the right boundary of the submatrix

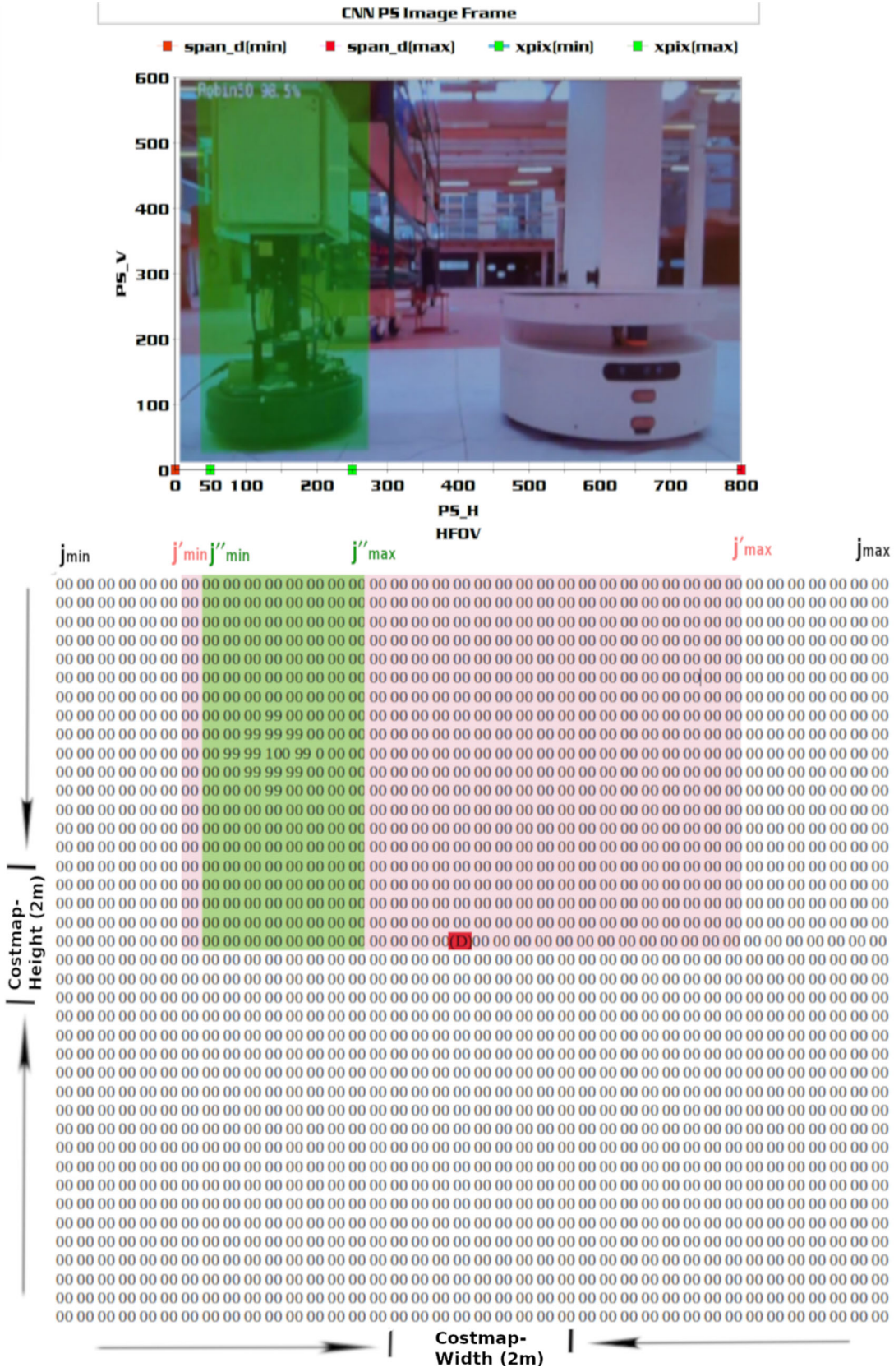


Fig. 3 Pixel to Cost map RoS translation

marking the last column position of the submatrix. The submatrix representing the effective local costmap that matches the size of the camera frame span will only be considered as the region of search (RoS). In Fig. 3, an example that visualizes the submatrix and is highlighted in pink can be seen. The submatrix as shown is aligned with the image frame above.

From knowing $j'_{\min}$ and $j'_{\max}$, the width of the submatrix which is also equal to the HFOV span in meters represented as span_d is calculated. Given the map resolution, resol (m / cell), of the local costmap by the user that is used as an input parameter to construct the local costmap, it is possible to calculate the minimum span limit $\text{span}_{d\min}$, and maximum span limit $\text{span}_{d\max}$ in meters shown in Eqs. 1 and 2.

$$\text{span}_{d\,\min} = j'_{\min} * \text{resol} \tag{1}$$

$$\text{span}_{d\,\max} = j'_{\max} * \text{resol} \tag{2}$$

$$\text{span}_d = j'_{\min} + j'_{\max} \tag{3}$$

The RoS column range RoS_{CR} can be calculated in Eq. 4.

$$\text{RoS}_{\text{CR}} = j'_{\max} - j'_{\min} \tag{4}$$

which represents the column size of the submatrix.

To align the horizontal pixels of the HFOV to the submatrix, in Eq. 5 the algorithm calculates the NPC, which represents how many pixels of the PS H axis are associated per cell of the submatrix (pixels/cell).

$$\text{NPC} = \frac{PS\_H}{\text{RoS}_{\text{CR}}} \tag{5}$$

Using $x\text{pix}_{\min}$ and $x\text{pix}_{\max}$ and knowing that the robot is at the center of its local costmap$c_{\frac{i_{\max}}{2}\,\frac{j_{\max}}{2}}$, which is marked as the red square in the middle of the matrix in Fig. 3, the algorithm calculates $j'_{\min}$ and $j'_{\max}$, which represents the left and right column position of the sub-submatrix as shown in Eqs. 6 and 7.

$$j''_{\min} = \frac{x\text{pix}_{\min}}{\text{NPC}} + j'_{\min} \tag{6}$$

$$j''_{\max} = \frac{x\text{pix}_{\max}}{\text{NPC}} + j'_{\min} \tag{7}$$

This sub-submatrix C'' therefore that contains the cells $C''_{ij''}$, which represents the boundary box position in the effective costmap in the entire local costmap.

Figure 3 shows an example of having a 2 m × 2 m local costmap C with elements c [40], where a the output image frame from the CNN model with a PS (PS_H 800 pixels, PS_V 600 pixels) is shown. The left and right boundaries of the detected boundary box are $x\text{pix}_{\min} = 50$ and $x\text{pix}_{\max} = 250$ as shown in the image. The entire local costmap C can be written as Eq. 8.

$$C = \begin{bmatrix} c_{00} & c_{01} & c_{02} & \cdots & c_{0j_{\max}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{i_{\max}0} & c_{i_{\max}1} & c_{i_{\max}2} & \cdots & c_{i_{\max}j_{\max}} \end{bmatrix} \tag{8}$$

where $i_{\max}$ and $j_{\max} = 40$. The *Detector* however, is at the position of $i = \max/2$ and $j = \max/2$. The range of the RoS for the rows is from 0 to $i = \max/2$. Applying the equations above, it is possible to visualize the effective costmap in Fig. 3 as all the elements of the matrix colored in pink and green, which can be written as the submatrix c' in Eq. 9.

$$C' = \begin{bmatrix} c'_{0j'_{\min}} & c'_{0(j'_{\min}+1)} & \cdots & c'_{0j'_{\max}} \\ \vdots & \vdots & \ddots & \vdots \\ c'_{i\frac{\max}{2}j'_{\min}} & c'_{i\frac{\max}{2}(j'_{\min}+1)} & \cdots & c'_{i\frac{\max}{2}j'_{\max}} \end{bmatrix} \tag{9}$$

The boundary box RoS which is shown in Fig. 3 as all the elements of the array colored in green, which can be written as the sub-submatrix c'' in Eq. 10.

$$C'' = \begin{bmatrix} c''_{0j''_{\min}} & c''_{0(j''_{\min}+1)} & \cdots & c''_{0j''_{\max}} \\ \vdots & \vdots & \ddots & \vdots \\ c''_{i\frac{\max}{2}j''_{\min}} & c''_{i\frac{\max}{2}(j''_{\min}+1)} & \cdots & c''_{i\frac{\max}{2}j''_{\max}} \end{bmatrix} \tag{10}$$

The cell $c''_{ij''}$ with a cost value of 100 will represent the *Pawn* location in the *Detector* frame. The coordinates of the Pawn, therefore, can be directly extracted and recorded.

4.3 Third part: handshake

Up to this stage, only the coordinates of the *Pawn* in the *Detector* map-frame alone are known. However, to able the robots to collaborate and share important location information as we will discuss in Sect. 6, the orientation of the *Pawn* MUST be known. The relative orientation of both robots cannot be directly extracted using ROS transforms (TF packages [24]). This is due to that the transform trees or the frames of both robots are not connected. More input is needed to compute the relative pose orientation of the *Pawn*. Therefore, a communication handshake between the *Detector* and the *Pawn* will take place. This handshake is done through a simple ROS service, "peer-to-peer communication," where an exchange of information is done between the robots. Since the coordinates of the *Pawn* $\overrightarrow{r}_P = (x_P, y_P)$ in the *Detector* map-frame are known, the distance between the *Detector* and the *Pawn* can be calculated using Eq. 11.

$$d = \|\vec{r}_P - \vec{r}'_D\| \tag{11}$$

where $\vec{r}'_D = (x'_D, y'_D)$ is the coordinates of the *Detector* in its own map frame. This calculated distance, will be then sent using the ROS service to the *Pawn*. In return, requesting an exchange of information from the *Pawn*. This information data will contain its own coordinates, hence, the *Detector*'s coordinates in the *Pawn* map-frame. Given the distance between the *Detector* and the *Pawn*, the algorithm within the *Pawn*, will search in its map-frame for a cell with a cost factor $c_{ij} = 100$, which represents an obstacle within approximately a distance d from itself. The coordinates of that obstacle will be obtained. At this point, the coordinates of that obstacle/object having the same footprint of the *Pawn*, represent the coordinates of the *Detector* in the *Pawn*'s map-frame. Since both robots are not connected within their frame tree in ROS, meaning they are independent, these coordinates must be translated and cannot be used directly. In sub Sect. 4.4, the explication of how the *Pawn* pose orientation and the frame transforms are calculated will be presented.

4.4 Fourth part: processing and localizing

Once the *Detector* obtains the coordinates of the *Pawn* in the *Detector*'s frame, and the *Detector*'s coordinates in the *Pawn*'s frame, the algorithm, using rotation matrices as shown in the following equations, will be used to determine the relative pose. In ROS language, it would be the transform between the two frames, "*odom* − *pawn*" and "*odom* − *detector.*" Using Rviz [25], the illustration of the *Pawn* pose in x, y is shown, and the orientation in the z axis (*yaw*) in the local cost map. All experiments were conducted on a 3 × 3 meter scaled grid with 0.5-meter square-divisions as a ground truth in the laboratory and in Rviz. To further explain the algorithm that computes the relative pose we assume that:

$\vec{r}_D = (x_D, y_D)$ is the coordinates of the *Detector* with respect to the *Pawn* (in the *Pawn*'s map-frame).

$\vec{r}_P = (x_P, y_P)$ is the coordinates of the *Pawn* with respect to the *Detector* (in the *Detector*'s map-frame).

$\vec{r}_{D0} = (x_{D0}, y_{D0})$, is the coordinates of the *Detector* with respect to the *Pawn* having rotating the *Pawn* to equal orientation of the *Detector* (in the *Pawn* robot map-frame). Considering the above declarations, Eq. 12 can be concluded.

$$\vec{r}_{D0} = -\vec{r}_P \tag{12}$$

Having applied the rotation matrix formula between x_D, y_D and x_{D0}, y_{D0} to compute ϕ, which is the relative angle between the *Pawn* and the *Detector*, shown in Eq. 15. In Fig. 4, the illustration of the corresponding parameters and the frame transformation relation between the *Detector* and the *Pawn* can be seen.

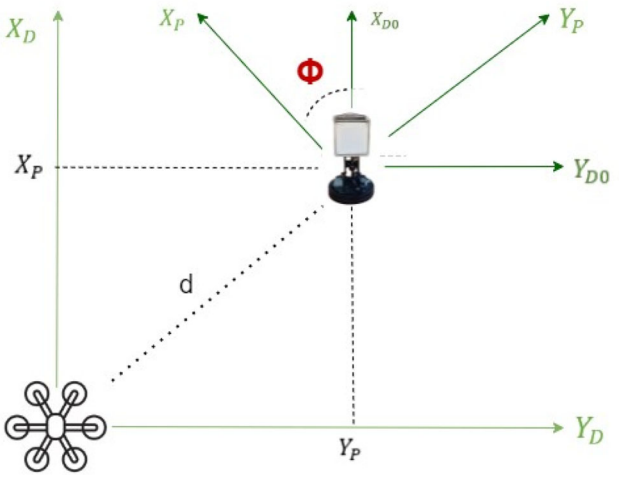


Fig. 4 The *Detector* and the *Pawn* frame transformation relation

$$\begin{bmatrix} x_D \\ y_D \end{bmatrix} = \begin{bmatrix} \cos\emptyset & \sin\emptyset \\ -\sin\emptyset & \cos\emptyset \end{bmatrix} \cdot \begin{bmatrix} x_{D_0} \\ y_{D_0} \end{bmatrix} \tag{13}$$

Which yields to Eq. 14,

$$\begin{bmatrix} x_D \\ y_D \end{bmatrix} = \begin{bmatrix} x_{D_0} & y_{D_0} \\ y_{D_0} & -x_{D_0} \end{bmatrix} \cdot \begin{bmatrix} \cos\emptyset \\ \sin\emptyset \end{bmatrix} \tag{14}$$

And Eq. 15.

$$\begin{bmatrix} \cos\emptyset \\ \sin\emptyset \end{bmatrix} = \begin{bmatrix} x_{D_0} & y_{D_0} \\ y_{D_0} & -x_{D_0} \end{bmatrix}^{-1} \cdot \begin{bmatrix} x_D \\ y_D \end{bmatrix} \tag{15}$$

Resulting in Eq. 16

$$\begin{bmatrix} \cos\emptyset \\ \sin\emptyset \end{bmatrix} = \frac{1}{x_{D_0}^2 + y_{D_0}^2} \cdot \begin{bmatrix} x_{D_0} \cdot x_D + y_{D_0} \cdot y_D \\ y_{D_0} \cdot x_D - x_{D_0} \cdot y_D \end{bmatrix} \tag{16}$$

The angle ϕ can be extracted from the following equations Eqs. 17 and 18.

$$\emptyset = \pm \arccos\left[\frac{x_{D_0} \cdot x_D + y_{D_0} \cdot y_D}{x_{D_0}^2 + y_{D_0}^2} \right] \tag{17}$$

By using Eq. 17 two different angles are extracted, in order to verify the correct one which represents which rotation direction, ϕ is to be calculated using Eq. 18 and find the matching angles.

$$\emptyset = \pm \arcsin\left[\frac{x_{D_0} \cdot x_D + y_{D_0} \cdot y_D}{x_{D_0}^2 + y_{D_0}^2} \right] \tag{18}$$

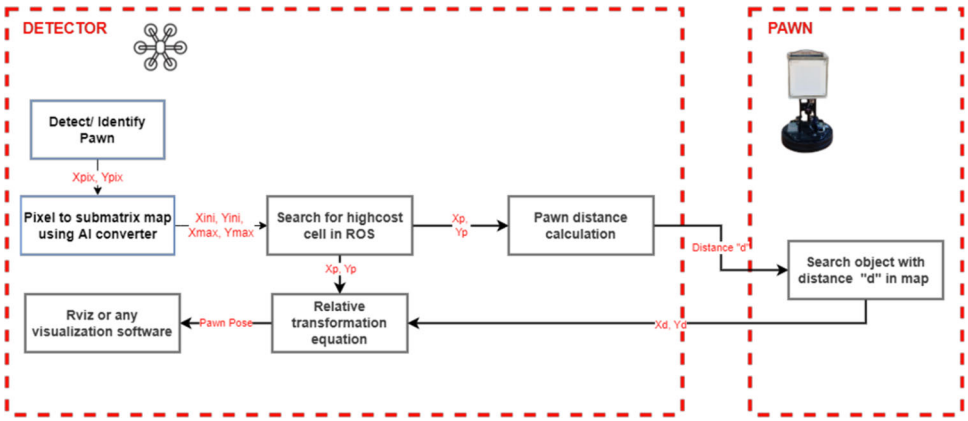


Fig. 5 DMLS method workflow and block diagram

A block diagram showing the workflow of the DMLS, and the full handshake between the *Detector* and the *Pawn*, is shown in Fig. 5.

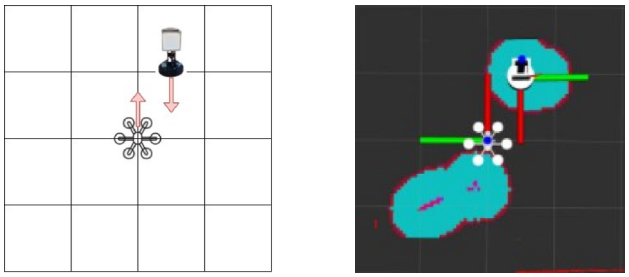
5 Comparison with QR-code pose estimation method

5.1 Scenario 1: pose estimation at $d = 1.00$ m

In order to validate the performance of our localization method, the results are compared with the ones from using the conventional QR-code pose estimation method for robotics [26]. In Scenario 1, we analyze and compare the results of both the DMLS and the QR-code localization methods on both robots, the *Detector* and *Pawn*. The *Pawn* is placed at a one-meter distance from the *Detector* in the x-axis, which represents the front-facing direction of the *Detector* map-frame. The *Pawn* was placed in 4 different positions in the y-axis, which were: $\overrightarrow{r}_P = (1.00\text{ m}, 0.65\text{ m})$, $\overrightarrow{r}_P = (1.00\text{ m}, 0.25\text{ m})$, $\overrightarrow{r}_P = (1.00\text{ m}, -0.25\text{ m})$, $\overrightarrow{r}_P = (1.00\text{ m}, -0.65\text{ m})$. At each position, 4 different sub-experiments are conducted. In each, the *Pawn* is oriented to a different relative orientation: $\phi = 0^\circ, 90^\circ, -90^\circ, 180^\circ$, ending up with a total of 16 sub-experiments. The tests were executed on a 3 m × 3 m grid with 0.5 m × 0.5 m subdivisions in the laboratory and in *Rviz*. The *Detector* was in a static position with a fixed orientation for all the tests.

5.1.1 Experiment 1A, Scenario 1, using DMLS

In this experiment, the DMLS is used for localizing and relative orientation measuring of the *Pawn* in the *Detector* map-frame for all of the 16 sub-experiments of Scenario 1. In Fig. 6, an example of having the *Pawn* positioned at $\overrightarrow{r}_P = (1.00\text{ m}, 0.65\text{ m})$ and $\phi = 180^\circ$ is shown. Figure 6a shows the sketch of the scenario done in the laboratory, and Fig. 6b shows an *Rviz* illustration of the local costmap of the *Detector*



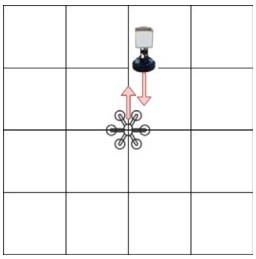
(a) Sketch of the scenario from the laboratory. (b) *Rviz* illustration of the local costmap observation by the *Detector*.

Fig. 6 Scenario 1, Experiment 1A. *Pawn* positioned at $\overrightarrow{r}_P = (1.00\text{ m}, 0.65\text{ m})$, $\phi = 180^\circ$ from *Detector*

used to obtain an accurate estimation of the position and orientation of the *Pawn* within the *Detector* map-frame. The red axis indicates the direction each robot is facing ("x-axis").

5.1.2 Experiment 1B, Scenario 1, using QR-pose estimation method

In Experiment 1B, the QR-code pose estimation method is used for localizing the *Pawn* in the *Detector* map-frame for all of the 16 sub-experiments of Scenario 1. In Fig. 7, an example where the *Pawn* is positioned at $\overrightarrow{r}_P = (1.00\text{ m}, 0.25\text{ m})$ and $\phi = 180^\circ$ is shown. Figure 7a shows the sketch of the scenario done in the laboratory, and Fig. 7b shows an image of the detected QR-code and corresponding pose estimation. The results are presented in Table 1, which shows that from the 16 sub-experiments, only in 8 the QR-code pose estimation method was able to get a successful detection of the *Pawn* by the *Detector*. The QR-code pose estimation method failed to identify and read the QR-code when it was not entirely visible for the camera's field of view (FOV). This shows that for having a pose estimation, it is vital for the QR-code to fully appear in the captured camera frame, whereas for the



(a) Sketch of the scenario in laboratory.

(b) Screenshot of the *Detector* the detecting a QR code on the *Pawn*.

Fig. 7 Scenario 1, Experiment 1B. *Pawn* positioned at $\overrightarrow{r}_P$ = (1.00 m,0.25 m), $\phi = 180°$ from *Detector*

Table 1 Results of the experiments in Scenario 1 using the DMLS and QR-pose estimation methods

Pawn pos	Lab ϕ	DMLS ϕ	QR-code ϕ
$\overrightarrow{r}_P$= (1.00 m,0.65 m)	180°	180°	–
$\overrightarrow{r}_P$= (1.00 m,0.65 m)	-90°	-93°	–
$\overrightarrow{r}_P$= (1.00 m,0.65 m)	90°	90°	–
$\overrightarrow{r}_P$= (1.00 m,0.65 m)	$0°$	$0°$	–
$\overrightarrow{r}_P$= (1.00 m,0.25 m)	180°	180°	179°
$\overrightarrow{r}_P$= (1.00 m,0.25 m)	-90°	-81°	-90°
$\overrightarrow{r}_P$= (1.00 m,0.25 m)	90°	96°	86°
$\overrightarrow{r}_P$= (1.00 m,0.25 m)	0°	0°	-3°
$\overrightarrow{r}_P$= (1.00 m, −0.25 m)	180°	180°	178°
$\overrightarrow{r}_P$= (1.00 m, −0.25 m)	-90°	-81°	93°
$\overrightarrow{r}_P$= (1.00 m, −0.25 m)	90°	92°	93°
$\overrightarrow{r}_P$= (1.00 m, −0.25 m)	0°	0°	2°
$\overrightarrow{r}_P$= (1.00 m, −0.65 m)	180°	180°	–
$\overrightarrow{r}_P$= (1.00 m, −0.65 m)	-90°	-85°	–
$\overrightarrow{r}_P$= (1.00 m, −0.65 m)	90°	99°	–
$\overrightarrow{r}_P$= (1.00 m, −0.65 m)	0°	0°	–
Mean		$\mu = 3.643$	$\mu = 2.229$
Variance		$\rho^2 = 13.274$	$\rho^2 = 1.536$

DMLS this is not a restriction, as long as the CNN is able to identify the *Pawn*, and a pose estimation using.

DMLS can be accomplished by only the partial appearance of the *Pawn* in the camera FOV. Among all the results using the QR-pose estimation method, only those that were successful were used to calculate the absolute mean error, and its variance, in Table 1.

5.2 Scenario 2: Pose estimation at $d = 1.40$ m

In Scenario 2, to further test the range of detection with this setup, the same methodology is repeated as in Scenario 1, but this time increasing the distance between both robots to 1.40m, in the front axis of the *Detector* (x-axis), since it was observed that the QR-code method did not consistently work with greater distances. For this experiment, the *Pawn* is also placed in 4 different positions from the *Detector*, which were: $\overrightarrow{r}_P = (1.40\,m, 0.80\,m)$, $\overrightarrow{r}_P = (1.40\,m, 0.25\,m)$, $\overrightarrow{r}_P = (1.40\,m, -0.25\,m)$, $\overrightarrow{r}_P = (1.40\,m, -0.75\,m)$. At each position, experiments were conducted while placing the *Pawn* at 4 different orientations, obtaining 16 sub-experiment results in total.

5.2.1 Experiment 2A, Scenario 2, Using DMLS

In Experiment 2A, Experiment 1A in Scenario 1 is repeated but in Scenario 2. We conclude from the results of all obtained poses and the absolute mean error shown in Table 2, that at a higher distance, the DMLS method achieved to localize the *Pawn* in all 16 sub-experiments while maintaining almost the same performance.

5.2.2 Experiment 2B, Scenario 2, Using QR-pose estimation method

In Experiment 2B, Experiment 1B of Scenario 1 is repeated but in Scenario 2. Since the QR-code cannot be read when the QR-code is only partially in the FOV of the camera, only 8 of the 16 sub-experiments were successfully executed. The pose estimations and the absolute mean error of the QR-pose estimation method are shown in Table 2.

5.3 Scenario 3: pose estimation in case of different robots in range

In scenario 3, since our robots are expected to operate within the presence of other team member robots of different types, forms, and sizes also in more complex environments, the experiments done in Scenario 1 and 2 are repeated but this time involving an extra robot, known as *Adversary Robot*. The *Adversary Robot* has a size and footprint bigger than the *Pawn*, meaning that in some Scenarios, it could partially or fully distort the detection of the *Pawn* by blocking the LoS between the *Detector* and the *Pawn*.

Table 2 Results of the experiments in Scenario 2 using the DMLS and QR-pose estimation methods

Pawn pos	Lab ϕ	DMLS ϕ	QR-code ϕ
$\overrightarrow{r}_P$ = (1.40 m,0.8 m)	180°	180°	–
$\overrightarrow{r}_P$ = (1.40 m,0.8 m)	− 90°	− 81.41°	–
$\overrightarrow{r}_P$ = (1.40 m,0.8 m)	90°	92.79°	–
$\overrightarrow{r}_P$ = (1.40 m,0.8 m)	0°	0°	–
$\overrightarrow{r}_P$ = (1.40 *m*,0.25 *m*)	180°	180°	173.551°
$\overrightarrow{r}_P$ = (1.40 m,0.25 m)	− 90°	− 81.334°	− 93.63°
$\overrightarrow{r}_P$ = (1.40 m,0.25 m)	90°	96.379°	81.674°
$\overrightarrow{r}_P$ = (1.40 m,0.25 m)	0°	0°	− 1.757°
$\overrightarrow{r}_P$ = (1.40 m, − 0.25 m)	180°	180°	177.957°
$\overrightarrow{r}_P$ = (1.40 m, − 0.25 m)	− 90°	− 91.32°	− 91.584°
$\overrightarrow{r}_P$ = (1.40 m, − 0.25 m)	90°	89.088°	84.119°
$\overrightarrow{r}_P$ = (1.40 m, − 0.25 m)	0°	0°	− 7.248°
$\overrightarrow{r}_P$ = (1.40 m, − 0.75 m)	180°	173.38°	–
$\overrightarrow{r}_P$ = (1.40 m, − 0.75 m)	− 90°	− 83.25°	–
$\overrightarrow{r}_P$ = (1.40 m, − 0.75 m)	90°	98.213°	–
$\overrightarrow{r}_P$ = (1.40 m, − 0.75 m)	0°	0	–
Mean		$\mu = 3.673$	$\mu = 2.515$
Variance		$\rho^2 = 13.493$	$\rho^2 = 6.326$

5.3.1 Scenario 3.1, Robots on different sides of the Detector's FOV

In Scenario 3.1, the *Pawn* and *Adversary Robot* are placed at each side and infront of the *Detector*'s FOV. This concludes that the *Detector* would visually sense one robot at each side of the image camera frame. Figure 8 illustrates the scenario. The positions of all robots in the *Detector*'s map for scenario 3.1 can be found in Table 3.

Experiment 3.1A, Scenario 3.1, Using DMLS In Experiment 3.1A in Scenario 3.1, the DMLS is used for localizing the *Pawn* in the *Detector*'s map-frame. It can be clearly seen from Fig. 8a and the results shown in Table 4 that the *Detector* has successfully identified, distinguished, and localized the *Pawn* from the *Adversary Robot*.

Experiment 3.1B, Scenario 3.1, Using QR-code pose estimation method In Experiment 3.1B in Scenario 3.1, the QR-code pose estimation method is used to estimate the pose for the *Pawn*. It can be clearly seen from Fig. 8b and the results shown in Table 4 that the *Detector* has successfully identified, distinguished, and localized the *Pawn* from the *Adversary Robot*. The reason was that the QR-code was

(a) Detecting the *Pawn* using the DMLS method.

(b) Detecting the *Pawn* using the QR-pose estimation method.

Fig. 8 Scenario 3.1, Experiment 3.1A and 3.1B. The *Pawn* and *Adversary Robot* placed at each side of the *Detector*'s FOV

Table 3 Relative position of the *Detector*, *Pawn*, and *Adversary* robots in Scenarios 3.1, 3.2, and 3.3 with respect to the *Detector*

Scenario	$\overrightarrow{r}_D$	$\overrightarrow{r}_P$	$\overrightarrow{r}_A$
3.1	(0.00,0.00)	(1.00 m, − 0.25 m)	(1.00 m,0.25 m)
3.2	(0.00,0.00)	(1.00 m, − 0.25 m)	(1.50 m,0.00)
3.3	(0.00,0.00)	(1.00 m, − 0.25 m)	(0.50 m, − 0.25 m)

Table 4 Provides the results of Experiments in scenario 3.1 and 3.2 using the DMLS and QR-code pose estimation method

Scenario	Real ϕ	DMLS ϕ	QR-code ϕ
3.1	180°	180°	179°
3.2	180°	180°	180°

(a) Detecting the *Pawn* using the DMLS method.

(b) Detecting the *Pawn* using the QR-pose estimation method.

Fig. 9 Scenario 3.2, Experiment 3.2A and 3.2B. The *Pawn* and *Adversary Robot* placed at one side of the *Detector*'s FOV while the *Pawn* is closer to the *Detector*

fully visible to the *Detector*, which was able to obtain a pose of the QR-code.

5.3.2 Scenario 3.2, Robots positioned on the same side of the Detector's FOV

In Scenario 3.2, the *Pawn* and the *Adversary Robots*, are placed on the same side of the map. However, the *Pawn* was placed closer to the *Detector* than the *Adversary Robot*, as shown in Fig. 9. The positions of all robots for Scenario 3.2 can be found in Table 3.

Experiment 3.2A, Scenario 3.2, Using DMLS In Experiment 3.2A, the DMLS method is used for localizing the *Pawn* in the *Detector*'s map-frame. It can be clearly noted from Fig. 9, and the results shown in Table 4, that the *Detector*, has successfully identified, distinguished, and accurately localized the *Pawn* from the *Adversary Robot*, even though more cells had a high-cost value in the RoS submatrix.

Experiment 3.2B, Scenario 3.2, Using QR-pose estimation method In Experiment 3.2B, the QR-code pose estimation method is used for localizing the *Pawn* in the *Detector*'s map-frame. It can be clearly noted from Fig. 9b and the results shown in Table 4, which *Detector*, has successfully identified, distinguished, and accurately localized the *Pawn* from the *Adversary*.

(a) Fail detection of the *Pawn* using the DMLS method.

(b) Fail detection of the *Pawn* using the QR-code pose estimation method.

Fig. 10 Scenario 3.3, Experiment 4A and 4B. The *Pawn* and *Adversary Robot* placed at one side of the *Detector*'s FOV while the *Pawn* is hidden from the *Detector*

5.3.3 Scenario 3.3, Robots placed on the same side, with different relative distances to the Detector

In Scenario 3.3, the *Pawn* and the *Adversary Robot*, are placed on the same side of the *Detector*'s FOV, but this time the *Pawn* was behind the *Adversary Robot*, hindering its visualization and detection of the *Detector*, making it non-LoS.

The Experiments in Scenario 3.3, as shown in Fig. 10 failed, as none of the methods could detect the *Pawn*. In Fig. 10a, an illustration of how the DMLS fails to detect the *Pawn* is shown, while Fig. 10b shows the QR-code pose estimation method failing to detect the QR-code.

5.4 5.4 Scenario 4: continued pose detection in 360°

The aim of the Experiment in scenario 4 is to test the detection robustness of both methods. For doing so the *Pawn* is continuously rotated along its center (z-axis) with a pre-set angle in the CW direction. For each rotation, both methods are tested to verify if they were able to achieve a successful detection of the *Pawn*. In total, 16 different sub-experiments were tested on both methods. The *Pawn* was placed at $d = 1.00m$ in the x-axis from the *Detector*. The locations of the *Pawn* and the *Detector* can be found in Table 5.

Table 5 Relative position of each robot in Scenario 4 to the *Detector* map-frame

$\vec{r}_D$	$\vec{r}_P$
(1.00 m,0.00 m)	(1.00 m,1.00 m)

Table 6 Provides the results of Experiments in Scenario 4 using the DMLS and QR-pose estimation method

Real ϕ	DMLS ϕ	QR-code ϕ
0°	0°	359°
30°	30°	28°
45°	46°	41°
60°	56°	62°
90°	84°	93°
120°	126°	127°
135°	140°	138°
150°	155°	147°
180°	180°	178°
210°	215°	206°
225°	229°	224°
240°	240°	236°
270°	270°	268°
300°	297°	311°
315°	317°	319°
330°	335°	328°
Mean	$\mu = 2.898$	$\mu = 3.599$
Variance	$\rho^2 = 5.30$	$\rho^2 = 5.754$

5.4.1 Experiment 4A, Scenario 4, Using DMLS

In Experiment 4A, the DMLS for is used for localizing the *Pawn* in the *Detector*'s map-frame for all 16 sub-experiments of Scenario 4. It can be clearly noted from the results shown in Table 6 that the *Detector* has successfully detected and accurately localized the *Pawn* in all 16 orientations.

5.4.2 Experiment 4B, Scenario 4, Using QR-pose estimation method

In Experiment 4B, the QR-code pose estimation method was used for localizing the *Pawn* in the *Detector*'s map-frame for all 16 sub-experiments of Scenario 4. It can be clearly noted from the results shown in Table 6 that the *Detector* has successfully detected and accurately localized the *Pawn* in all 16 orientations.

It is important to note that both methods were able to recover after a faulty detection and successfully detect and localize the *Pawn* when the parameters that affected the detection were present. Table 7 shows a fault tolerance comparison for both methods with different conditions:

From Table 7, we can see that both models require different environmental conditions to operate as needed. For example, the DMLS requires that the *Detector* and the *Pawn* are able to connect through a wireless network, whereas the QR-code pose estimation method does not, however, the latter is sensitive to the relative orientation of the QR-code as it fails to detect the QR-code at some angles and also requires modification of the environment in order to install QR-codes. Finally, both methods require that both robots are in LoS to able to work properly.

6 Applications

6.1 Laboratory experiment

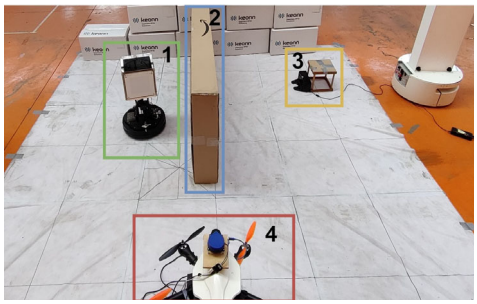
The docking station and the robot docking mechanism are designed to work together, providing a mechanical and electrical connection between the charging system and the robot [27]. This communication in our case is composed of infrared LoS transmitted signals from the docking sensors on the *Pawn*, with the receiver on the *Docking Station*. Taking advantage of the full capacity of the CNN used by the DMLS, the CNN model was trained to detect multiple important entities in the environment, one of them is the *Docking Station*. We simulate a scenario of the *Pawn*, during an end of a mission in an un-explored environment, where it starts searching for the *Docking Station*, in order to self-charge and continue toward task completion. However, the *Pawn* fails to detect the *Docking Station* due to an obstacle blocking the LoS communication between the *Pawn* and the *Docking Station* as shown in Fig. 11a. The *Detector*, however, happens to detect both the *Pawn* and the *Docking Station* using the pre-trained CNN model on board. Utilizing the computed *Pawn* pose using DMLS, the *Detector* can share with the *Pawn* the exact pose of the *Docking Station* in the *Pawn*'s map-frame.

In Fig. 11a and b, the entity tagged as "1" is the *Pawn*. The entity tagged as "2" is the obstacle blocking the LoS between

Table 7 Shows fault tolerance comparison between both methods in different situations

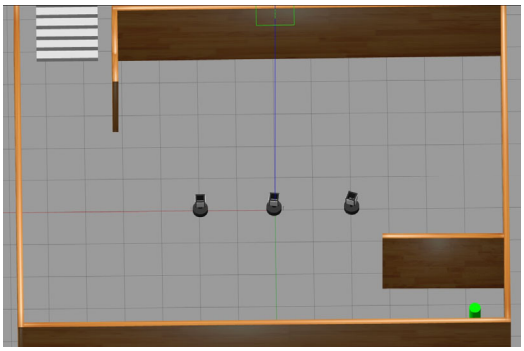
Method	Network communication between Robots	Robot relative orientation	Modification of the environment	LoS
DMLS	Not tolerant	Tolerant	Tolerant	Not tolerant
QR-code	Tolerant	Not tolerant	Not tolerant	Not tolerant

(a) POV of the *Detector* during the test.

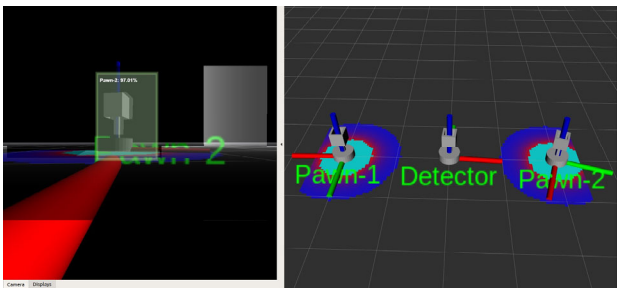


(b) Scenario picture of the test

Fig. 11 Laboratory illustration of the Application scenario having Multi-robots collaboration using DMLS



(a) Gazebo illustration of the Scenario.



(b) Rviz Detection of Pawn-1 & 2.

Fig. 12 Simulation Scenario 1: The Detector localizing and sharing the positions to all robots

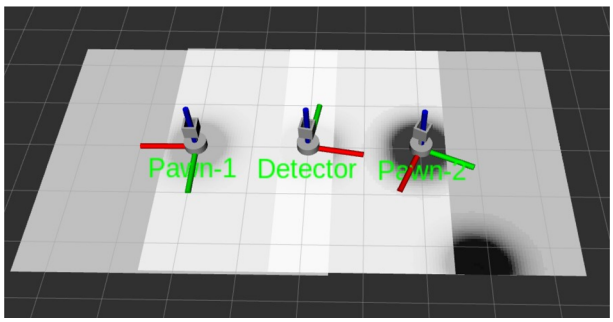


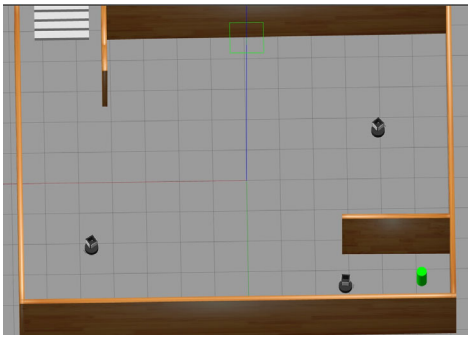
Fig. 13 Simulation Scenario 1: An illustration from the Detector's frame showing local costmaps from all robots

the *Docking Station* and the *Pawn*. The entity tagged as "3" is the *Docking Station*. The entity tagged as "4" is the *Detector* which cannot be seen in Fig. 11a since it is the source point. It can also be seen how both *Pawn* and *Docking Station* are being identified and detected with high level of confidence.

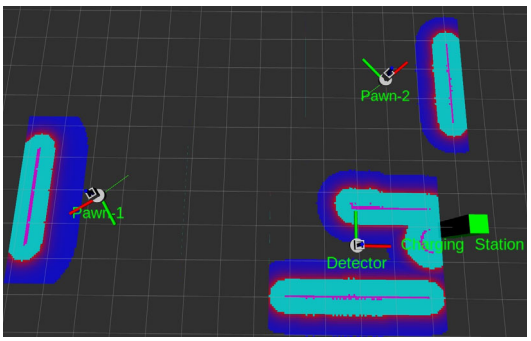
6.2 Simulation experiments

6.2.1 Feature location sharing using vision

In Simulating Scenario 1, Gazebo simulation platform is used for simulating Scenario 1 that is shown in Fig. 12a, a Gazebo world model is designed for the purpose of showing how a group of UGVs using the DMLS, can collaborate to share important features in the environment. For this Scenario, 3 UGVs will be used, one *Detector* and 2 *Pawns* called *Pawn-1* and *Pawn-2*. The UGVs will initially spawn 1m away from each other. The *Detector*'s initial position would be in the middle and the rest at the sides. First, the *Detector* will detect *Pawn-1*, due to the fact that it is in LoS with the *Detector*, lays within its local costmap detection region, and is detected by the CNN. Then the *Detector* does a 360deg. rotation to check if it is able to detect more robots in the area. As a result, *Pawn-2* will also be detected and successfully localized using the DMLS. Figure 12b shows the detection of both *Pawns*. The left side of the image shows the camera output visually detecting the robots and the right shows what is able to be sensed in the *Detector's* frame. After successful detection and localization between the robots, all robots would be able to share their local costmaps with each other as shown in Fig. 13. This will increase the environmental awareness of each robot using the help of the others. The robots will then pursue the tasks that they are designed for. In the case of this Scenario, the robots will move in random directions exploring the map. In Fig. 14a and b, the *Detector* has discovered and successfully located important features in the environment. Not only features that are beneficial to the robots like the charging/docking station are shared when discovered and

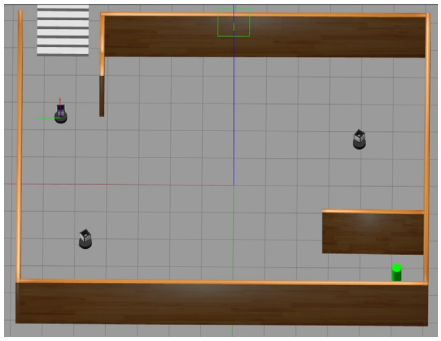


(a) Gazebo illustration of Localizing theDocking station.

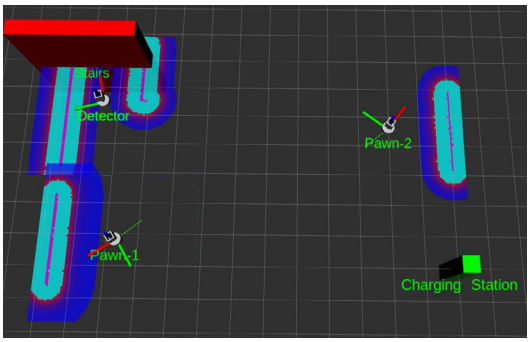


(b) Rviz illustration of Localizing theDocking station.

Fig. 14 Simulation Scenario 1: Localizing and sharing the docking station position to all robots

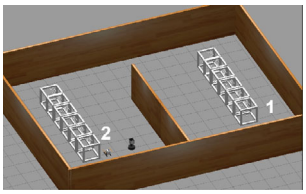


(a) Gazebo illustration of Localizingthe Stairs.

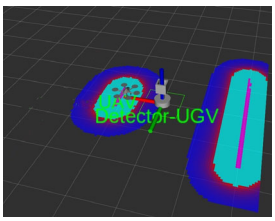


(b) Rviz illustration of Localizing the Stairs.

Fig. 15 Simulation Scenario 1: Localizing and sharing the stairs position to all robots



(a) Gazebo illustration of Scenario Simulation Scenario 2



(b) Rviz illustration Detection of Pawn.

Fig. 16 Simulation Scenario 2: The Detector localizing the Pawn

localized, however, using the DMLS robots can warn each other of dangerous zones. An example of localizing surfaces unsuitable for the robots to navigate through (i.e., stairs) is shown in Fig. 15a and b. The *Detector* uses just the first part of the DMLS algorithm to detect and localize these features, as in this case the orientation of the objects is not important.

6.2.2 Collaborating in the existence of RFID tags

In Simulation Scenario 2, shown in Fig. 16a, a gazebo world model is designed to show that using the DMLS, heterogeneous inventory robots can collaborate in environments where RFID sensors exist for the purpose of autonomous mapless navigation and completing an inventory mission. The goal is to increase the overall team performance of completing an inventory mission in environments where products are represented by RFID tags, such environments are warehouses or retail shops. The robots will exploit the heterogeneity feature in the team for exploring new regions that have RFID tags. Inventory UGVs have proven to perform well during recent years [28]. They are robust machines that can carry heavy payloads which allows large power sources to be mounted onboard. This able them to operate for a considerable amount of time. However, these robots are slow, considered limited and not adequate for exploration purposes. On the other hand, UAVs are adequate for exploration purposes due to their agility, and maneuverability and are undependable of the ground surface type. However, UAVs suffer from low flight time and have weight and size constraints for the payload it can lift. This Scenario focuses on the exploitation of the benefits of both types of robots for completing an inventory task. Two heterogeneous robots are used. This time the *Detector* will be a UGV.

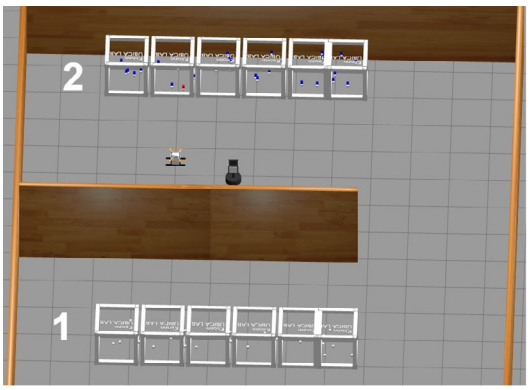


Fig. 17 Simulation Scenario 2: Gazebo illustration of the RFID tag detection located on the left shelves (2)

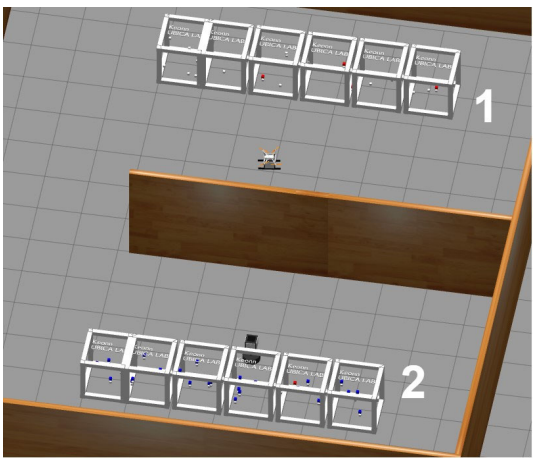


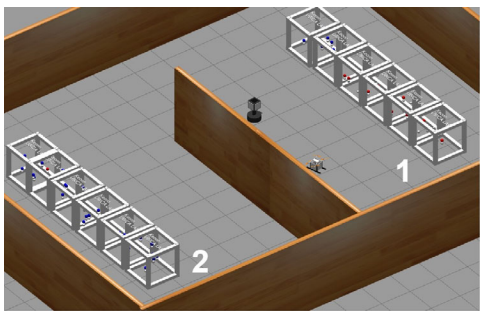
Fig. 18 Simulation Scenario 2: Gazebo illustration of the RFID tag detection located on the right shelves (1)

The *Pawn* will be a UAV. Both robots will be equipped with an RFID reader [29] and directional antennas [30]. The map will contain 2 shelves. Both placed far from each other and having a wall almost separating them as shown in Fig. 16a.

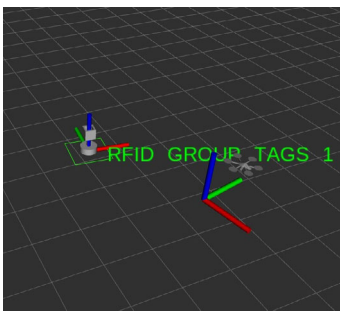
Each shelf contains 6 boxes, each box contains a group of RFID tags. The non-detected RFID tags will be represented as gray cubes. The RFID tags that are detected by the *Detector* would be represented as blue cubes. Finally, the RFID tags that are detected by the *Pawn* would be represented as red cubes. RFID technology is simulated using a ROS-based Gazebo plugin [31]. Since the *Detector* uses RFID stigmergic-based navigation [32], therefore it would only navigate in the region where it can detect new RFID tags. This implies that since the shelf on the right side marked as (1) in Fig. 16a, would be considered outside of the detection range, the *Detector* would not be aware of the existence of the RFID tags on that side. It will only read the RFID tags on the left side marked as (2) in Fig. 16a and shown in Fig. 17.

Since the *Pawn* operates as a "scout," it continuously searches for new zones where populations of RFID tags exist and share the location of where it was able to detect these new RFID tags. The *Pawn* uses an autonomous navigation scheme, which moves in a chosen direction as long as no obstacles are detected in its direction of movement. The *Pawn* will choose a different direction in case of an obstacle is detected in the direction of movement. In Fig. 18, more red cubes can be seen, which represents the detected RFID tags while the *Pawn* is flying through the environment. As soon as new RFID tags are detected by the *Pawn*, as shown in Figs. 18 and 19b, the location coordinates of where the new RFID tags are read will be relayed to the *Detector*. The *Detector* therefore will move to new regions where new tags are explored by the *Pawn*, each time it receives this data.

Fig. 19 Simulation Scenario 2: The Detector Moving toward the new explored zone of RFID tags, sent by the Pawn



(a) Gazebo illustration of the Detector and Pawn positioned at the new discovered RFID tags zone.



(b) Rviz illustration of the Detector and Pawn positioned at the new discovered RFID tags zone.

7 Conclusions

Driven by the growing interest in multi-robots trying to achieve collective intelligence from individual simplicity, a strategy based on ROS that is used to localize and measures the relative orientation for distributed multi-robots is presented, for the case where the team robots are not connected within their transform trees. This enables heterogeneous multi-robots to share valuable resources among themselves. The proposed technique does not require modification of the environment, such as placing QR-codes, beacons, or complex sensors, nor it requires prior knowledge of the map for it to function and obtain good results, as do most currently used localization methods. We also prove empirically that it is computationally inexpensive. It has been tested on several low computation power single board computers (SBC) such as JetsonNano and on different robot platforms. The proposed method relies only on the existing basic hardware of an autonomous robot in multi-robot's scenarios. Any robot that can use the basic navigation and path planning packages provided by ROS will be able to run this localization model. At least one robot in the multi-robots group equipped with a visual sensor, and all robots in the group must be able to communicate with each other.

Though the DMLS method, as illustrated in experiments, proves to have accurate results and good performance similar to the QR-code pose estimation method in most cases, it still has some limitations. These limitations are addressed in Section 8.

8 Future work

During intensive experimentation on our proposed localization method, some limitations and opportunities were discovered:

a) *Reducing the RoS in the map*, by improving the algorithm to compute a smaller RoS, which indicates the MLH obstacle to be the robot. This will significantly reduce the possibility of the *Detector* mis-locate the *Pawn*.
b) *Increasing the resolution of the local costmap cells*, by increasing the resolution of the map cells, and changing it from approximately 2.5 cm per cell to just a few millimeters per cell. This will lead to a higher resolution in extracting the exact position of the *Pawn*.
c) *Using AI to analyze the map*, designing a neural network (NN) whose input data consists of the local cost map of the robot by converting the map-frames to consecutive images. This data set will be fed into the NN during the detection phase. The goal is to predict and distinguish the robot shape in a map, from different shapes and patterns of different obstacles.
d) *Using 3D mapping to increase method's efficiency*, eliminating intermediate steps in Sect. 3, such as the Detector landing in order to be detected by the Pawn. Increasing the dynamism of the overall workflow of the DMLS Frame-work.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

1. Cox I (1991) Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. IEEE Trans Robot Autom 7(2):193–204. https://doi.org/10.1109/70.75902
2. Chenavier F, Crowley J (1992) Position estimation for a mobile robot using vision and odometry. In: Proceedings 1992 IEEE international conference on robotics and automation, vol.3, pp. 2588–2593. https://doi.org/10.1109/ROBOT.1992.220052
3. Barker A, Brown D, Martin W (1995) Bayesian estimation and the kalman filter. Comput Math Appl 30(10):55–77
4. Lu F, Milios E (1994) Robot pose estimation in unknown environments by matching 2d range scans. In: 1994 Proceedings of IEEE conference on computer vision and pattern recognition, pp. 935–938. https://doi.org/10.1109/CVPR.1994.323928
5. Parker L (2000) Current state of the art in distributed autnomous mobile robotics. Distrib Auton Robot Syst 4(01):3–14. https://doi.org/10.1007/978-4-431-67919-61
6. Casamayor-Pujol V, Morenza-Cinos M, Gastón B, Pous R (2020) Autonomous stock counting based on a stigmergic algorithm for multi-robot systems. Comput Ind 122:103259
7. Rosas-Cervantes VA, Hoang QD, Lee SG, Choi JH (2021) Multi-robot 2.5 d localization and mapping using a Monte Carlo algorithm on a multi-level surface. Sensors 21(13):4588. https://doi.org/10.3390/s21134588
8. Vidal-Calleja TA, Berger C, Solà J, Lacroix S (2011) Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain. Robot Auton Syst 59(9):654–674
9. Hausman K, Müller J, Hariharan A, Ayanian N, Sukhatme GS (2015) Cooperative multi-robot control for target tracking with onboard sensing. Int J Robot Res 34(13):1660–1677. https://doi.org/10.1177/0278364915602321
10. Leonard J, Durrant-Whyte H (1991) Mobile robot localization by tracking geometric beacons. IEEE Trans Robot Autom 7(3):376–382. https://doi.org/10.1109/70.88147

11. Rekleitis I, Dudek G, Milios E (2001) Multi-robot collaboration for robust exploration. Ann Math Artif Intell 31:7–40. https://doi.org/10.1023/A:1016636024246
12. Kato K, Ishiguro H, Barth M (1999) Identifying and localizing robots in a multi-robot system environment. In: Proceedings 1999 IEEE/RSJ international conference on intelligent robots and systems. Human and environment friendly robots with high intelligence and emotional quotients (Cat. No.99CH36289), vol. 2, pp. 966–971. https://doi.org/10.1109/IROS.1999.812805
13. Jennings C, Murray D, Little J (1999) Cooperative robot localization with vision-based mapping. In: Proceedings-IEEE international conference on robotics and automation, vol. 4, pp. 2659–2665. https://doi.org/10.1109/ROBOT.1999.773999
14. Fox D, Burgard W, Kruppa H, Thrun S (2000) A probabilistic approach to collaborative multi-robot localization. Auton Robot 8:325–344
15. Pereira GA, Kumar RV, Campos MF (2003) Localization and tracking in robot networks. Dep Papers (MEAM), p. 39
16. Stroupe A, Martin M, Balch T (2001) Distributed sensor fusion for object position estimation by multi-robot systems. In: Proceedings-IEEE international conference on robotics and automation, vol. 2, pp. 1092–1098
17. Gavrilut I, Tiponut V, Gacsadi A, Grava C (2007) Cnn processing techniques for multi-robot coordination. In: 2007 International symposium on signals, circuits and systems, vol. 1, pp. 1–4. https://doi.org/10.1109/ISSCS.2007.4292703
18. Alajami AA, Moreno G, Pous R (2022) Design of a uav for autonomous rfid-based dynamic inventories using stigmergy for mapless indoor environments. Drones 6(8):208
19. Sandler M et al (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4510–4520. https://doi.org/10.48550/arXiv.1801
20. Koubˆaa A et al (2017) Robot operating system (ROS). Springer, Berlin
21. Marder-Eppstein E (2018) MoveBase move base is a navigation package for robots. Accessed: 2022. [Online]. Available: http://wiki.ros.org/move base
22. Novotny F (2021) Costmap is a cell cost based map creator package for robots. http://wiki.ros.org/costmap2d. Accessed: 2021
23. Rowlands A (2017) Fundamental optical formulae. In: Physics of digital photography, ser. 2053–2563. IOP Publishing, Book Chapter, pp. 1–1 to 1–62.
24. TF transform package for tracking multiple coordinate frames over time. http://wiki.ros.org/tf. Accessed: 2022
25. Hershberger D (2022) Rviz a 3d visualization tool for ros. http://wiki.ros.org/rviz. Accessed: 2022
26. Novotny F (2021) QR pose estimator visp auto tracker is a qr code pose estimator package for robots. http://wiki.ros.org/costmap 2d?distro= noetic. Accessed: 2021
27. Song G, Wang H, Zhang J, Meng T (2011) Automatic docking system for recharging home surveillance robots. In: IEEE transactions on consumer electronics-IEEE Trans Consum Electron, vol. 57, pp. 428–435. https://doi.org/10.1109/TCE.2011.5955176
28. Morenza-Cinos M, Casamayor-Pujol V, Soler-Busquets J, Sanz JL, Guzm R, Pous R (2017) Development of an RFID Inventory Robot (AdvanRobot). Springer, Cham, pp 387–417
29. Keonn (2022) Keonn's AdvanReader 160 keonn's advanreader 160. https:// keonn.com/components-product/advanreader-160/. Accessed: 2022
30. Technologies K (2022) Keonn's AdvanReader SP11 keonn's advanantennasp11. https://keonn.com/components-product/advanantenna-sp11/. Accessed: 2022
31. Alajami AA (2022) Rfid sensors plugin for gazebo simulation tool in ros wiki. http://wiki.ros.org/RFIDsensor Gazebo plugin. Accessed: 2022
32. Heylighen F (2016) Stigmergy as a universal coordination mechanism I: definition and components. Cognit Syst Res 38:4–13. https://doi.org/10.13140/RG.2.1.4479.5044