Unified Indoor Mapping with Swarm Mobile Robot Based Monte Carlo Technique

Mohamed Yussuf, and Elsayed Abdelaziz
Depart. of Automotive Eng., Military Technical College, Egypt
{Mohammed1yusuf.awd, sayed.abdelaziz114}@gmail.com

Supervisor: Tamer Attia
Depart. of Automotive Eng., Military Technical College, Egypt

***Abstract*—This work aims to enable robots with the ability to map unknown indoor environments. The proposed technique is implemented with a differential drive robot (Ratbot). This work focuses on the mapping of indoor environment with a swarm of mobile robots. The navigation and mapping with a swarm of Ratbot are based on fusion of each data collected by each robot to build a unified map for the whole environment. The used sensors in this research are Light Detection and Ranging (LIDAR), IMU, and encoder to localize each robot and build the map of the environment. Based on communication network between thr Ratbots, the robots can share local sensing information to coordinate the actions of all robots in the network. The motion of all Ratbots is coordinated with a frontier exploration motion planner that has been augmented with new sampling Strategies. Finally, experimental results show the effectivness of the proposed technique in building a unified map with a swarm of mobile robot.**

I. Introduction

Indoor mapping is very difficult kind of mapping because we cannot use GPS. So we use the Simultaneous Localization and Mapping (SLAM) algorithm to solve this problem. In general the indoor mapping is very difficult because some environment have some long tunnels as underground subway.

Robotic swarm involves multi-robots in a team to solve a problem; each has its own task to serve the main purpose; sharing data using intercommunications. In addition, swarms have desirable properties such as self-organization, high redundancy, and the lack of single points of failure promote fault tolerance, scalability and flexibility [1].

Robotic swarm is a perfect solution to create a map for indoor environments because the whole map is divided into sub-maps which are assigned for each robot. So, we do not need single agent to move through these environments to map.

Mapping using single-robot is challenging enough but multiple robots adds another layer of challenging. In a multiple-robot environment, robots must share all available data to construct a global maps [2] but there are some problems while using robot swarm like relative poses of robots, uncertainty of the relative poses, updating maps with poses, and communications.

The key idea of markov localization is to compute a probability distribution over all possible locations in the environment. If the robot doesn't know its initial position the robot believe is uniformly distributed to reflect the global uncertainty. The robot maintains a belief distribution which is updated upon robot motion, and upon the arrival of sensor data. Such probabilistic representations are well-suited for mobile robot localization due to their ability to handle ambiguities and to represent degree-of-belief. [3].

In this study we use Monte Carlo Localization (MCL) [4] also known as particle filter localization, is an algorithm for robots to localize itself using a particle filter. Given a map of the environment, the algorithm estimates the position and orientation of a robot as it moves and senses the environment. The algorithm uses a particle filter to represent the distribution of likely states, with each particle representing a possible state, i.e., a hypothesis of where the robot is. The algorithm typically starts with a uniform random distribution of particles over the configuration space, meaning the robot has no information about where it is and assumes it is equally likely to be at any point in space. Whenever the robot moves, it shifts the particles to predict its new state after the movement. Whenever the robot senses something, the particles are resampled based on recursive Bayesian estimation, i.e., how well the actual sensed data correlate with the predicted state. Ultimately, the particles should converge towards the actual position of the robot.

In this paper, we introduce .

II. Autonomous Navigation and Mapping

A. Autonomous Navigation of Single Robot

Smooth and safe navigation of robot through the environment from start position to goal position while following safe path and producing the optimal path length, is the main aim of robot navigation. Regarding this matter, many techniques have been tackled for robot navigation and path planning. Robot navigation means the robot's ability to determine its position in its frame of reference and then to plan a path from its start position to its goal location. In order to navigate in its environment, the robot requires a map of the environment and the method in which to interpret this map.

Navigation can be defined as the combination of the three fundamental competences: Numbered (ordered) lists are easy to create:

1) Self-localization

$^{0}6^{th}$ IUGRC International Undergraduate Research Conference, Military Technical College, Cairo, Egypt, Sep. 5th – Sep. 8th, 2022

2) Path planning
3) Map-building

*1) **Methods of self-localization**:*

- **Kalman Filter** Kalman filtering has been successfully applied for mobile robot localization in many systems. But most of the time this method is unable to globally localize the robot because this approach can only represent one pose hypothesis resulting in the robot's inability to recover from possible total localization failures.
  The filter is based on the Bayes probability which assumes the model and uses this model to predict the current state from the previous state. Then, an error between the predicted value of the previous step and the actual measured present value obtained by measuring instrument is used to perform an update step of estimating more accurate state value. The filter repeats above process and increases the accuracy. This process is simplified as shown in Figure (1) However, the Kalman filter only applies to linear systems. Most of our robots and sensors are nonlinear systems and the EKF (Extended Kalman Filter) modified from Kalman filter are widely used

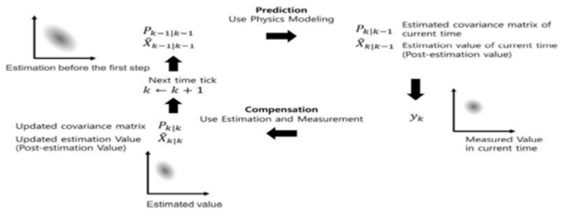
Prediction
Use Physics Modeling
$P_{k-1|k-1}$
$\hat{x}_{k-1|k-1}$
Estimation before the first step
$P_{k|k-1}$
Estimated covariance matrix of current time
$\hat{x}_{k|k-1}$
Estimation value of current time (Post-estimation value)
Next time tick
$k \leftarrow k+1$
Compensation
Use Estimation and Measurement
y_k
Updated covariance matrix
$P_{k|k}$
Updated estimation Value (Post-estimation Value)
$\hat{x}_{k|k}$
Measured Value in current time
Estimated value

Fig. 1. Basic concept of Kalman Filter

- **Monte Carlo localization** The main premise of Monte Carlo localization (MCL) is to represent p(l) by sets of n weighted samples (Li, Wi) Each Li corresponds to a robot position, and the Wi are positive numerical factors called importance weights, which add up to one. A procedure is used for The prediction and correction update of the sample sets. The MCL algorithm takes as input a sample set S representing the current position estimate p(l) , a sensor measurement s, and action information a,. Each sample representing the posterior distribution for p(l) is generated according to the following three steps
  - Resampling
  - Sampling
  - Importance sampling

*2) **Path planning**:* Path planning is one of the most essential part of autonomous mobile robot navigation system. Several methods have been developed to find solutions for the path planning problem. Path planning involves the determination of collision-free path from one point to another with the shortest possible associated path. Depending on the nature of environment, path planning can be divided into static and dynamic. If obstacles change their position with respect to time, it is referred as static path planning and if obstacles change their position and orientation with respect to time, then it is referred as dynamic path planning. There are several algorithms for path planning such as

1) roadmap
2) cell decomposition
3) potential fields
4) bug algorithms

We will be discussing potential fields method. Which showed the highest efficiency relative to other path planning algorithms for our case.

- **potential field method** as a particle represented by a point in configuration space q. The robot moves under the influence of an artificial field of forces produced by a goal and the obstacles present in the q. This field of forces is specified as the negative gradient of a potential function.
  The effect of the repulsive potential on the robot is clearly opposite from that of the attractive potential. now we want the obstacles to apply repulsive forces on the robot, where these forces are inversely Proportional in magnitude to the distance between the robot and each obstacle. On the other hand, repulsive forces from obstacles that are too far away from the robot to post significant danger are neglected. Therefore, a maximum effective distance of the obstacle from the robot is assigned. Any obstacle further than this effective distance will have no effect on the robot. Thus, the repulsive potential can be expressed as.

$$V_{attract}(q) = \frac{1}{2}K_{attraction}(q - q_{desired})^2 \tag{1}$$

$$F_{attract}(q) = K_{attraction}(q - q_{desired})^2 \tag{2}$$

$$V_{repul}(q) = \begin{cases} \frac{1}{2}K_{repul}(1/d - 1/d_o) & if \quad d < d_o \\ 0 & if \quad d \geqslant d_o \end{cases} \tag{3}$$

$$F_{repul}(q) = \begin{cases} -K_{repul}(1/d - 1/d_o)1/d^2 & if \quad d < d_o \\ 0 & if \quad d \geqslant d_o \end{cases} \tag{4}$$

*3) **Map building**:* The mapping and localizing is a chicken and egg problem. At certain points it is hard to consider carrying out what first, either the localization or mapping. Concurrently building map and localizing the robot in it is the best practice being followed. Many of the techniques are based on this concurrency and all try to find a solution for the problem of simultaneous localization of mapping. The context of mapping does not limit it extent only for mapping but it is also useful in path planning for a mobile robot in the environment. Once the proper map is produced the mobile robot feels easy to locate itself in some part of it and localize. But building a proper map is really a hard problem. The process of building a map goes through many phases as shown in Figure (1). Once the autonomous mobile robot is localized with reference to the world coordinates it should build a new map, if appropriate map is not available, or should start using a map.Mapping includes simultaneously estimating the pose of the robot and the map.

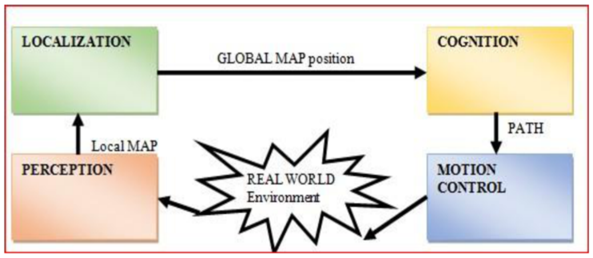

Fig. 2. building phases of map

Information Required in SLAM Let's look at the materials you need to create map with SLAM. First of all, we need to define what we need when creating a map as shown in Figure (3).**first** thing you need is the distance value. This meaning the robot being the center of measurement, the robot should be able to obtain the distance value from certain objects. For example, information such as "the chair is 2m away from the robot". Distance data scanned from the XY plane using sensors such as LDS and Depth camera is an example of such information.**Second** is the pose value which stands for the pose information of the sensor that is attached to the robot. Thus, the pose value of the sensor depends on the odometry of the robot. It is necessary to provide the odometry information to calculate the pose value. In the Figure below, the distance measured with Kinect is called 'scan' in ROS and the pose (position + orientation) information is affected by the relative coordinate, so it is called 'tf' (transform). As Figure below show, we run SLAM based on two pieces of information, 'scan' and 'tf', and create the map we want.

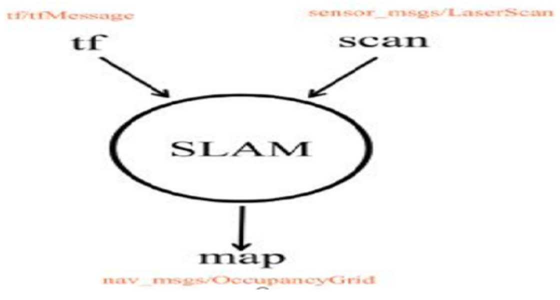

Fig. 3. data required in slam

using the platform Robot Operating System (ROS) that are available in Ubuntu. The algorithm chosen for this research is Gmapping. Gmapping used the Rao-Black wellized Particle Filter (RBPF) and take data from both laser sensor and robot pose to create a 2D grid map. as shown in Figure (4). the mapping flow process which is used for this research.

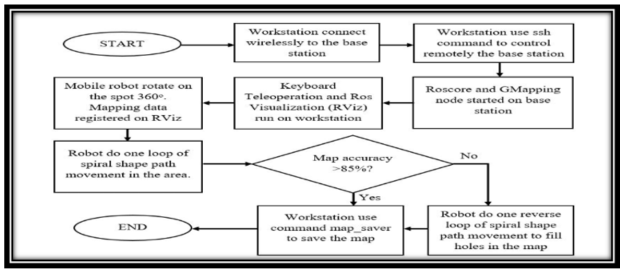
START
Workstation connect wirelessly to the base station
Workstation use ssh command to control remotely the base station
Roscore and GMapping node started on base station
Keyboard Teleoperation and Ros Visualization (RViz) run on workstation
Mobile robot rotate on the spot 360°. Mapping data registered on RViz
Robot do one loop of spiral shape path movement in the area.
Map accuracy >85%?
No
Yes
Robot do one reverse loop of spiral shape path movement to fill holes in the map
Workstation use command map_saver to save the map
END

Fig. 4. data

B. Swarm of robots make mapping to indoor environment

Multiple robots can cooperate to manipulate large objects, survey large areas in a short amount of time, and provide system redundancy to enable multiple robots to cooperate to do some missions better than single robot. The ratbot navigation will depend on the data which each ratbot shared With each other to build merged map by data which each ratbot Published about unknown environment. Within these networks, robots can share local sensing information and coordinate the actions of all robots in the network.This study hopes to use ratbots to excavate tunnels located in underground to help generate a 2-D map of them to protect people during their missions.

- **Control on multi-robots to move Autonomous** We need to make control on swarm. In this project, Swarm of robot is centralized control because one robot will be able to give other robots the goals that each robot needs to move autonomously to. first mission is a leader robot (robot 0)as shown in Figure (5) takes a point as a goal and all robot go there with shifting. Each robot takes a goal and makes Autonomous navigation to its modified goal.so we make network to connect all robots with each other to share data.
  In this project, we use three robots as a swarm. We give leader a goal and all robots move to make arrowhead shape at this goal.

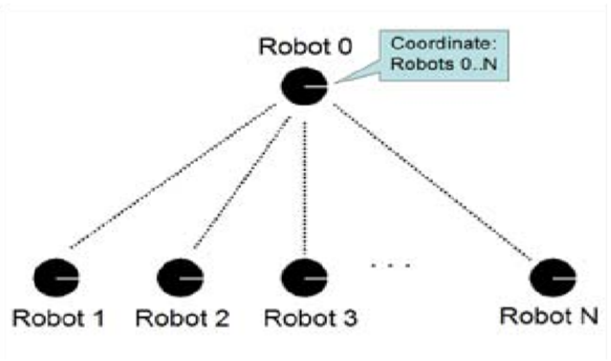

Fig. 5. robors coordinate

- **Frontier exploration** In order for the robots to autonomously explore an environment, we implemented frontier exploration from scratch. By manipulating, the occupancy grid generated from slam toolbox. we were able to find all the frontier edges and then group the edges into separate frontier regions via this algorithm. For each frontier region, we found the region's centroid.eventually, I determined which centroid was closest to the robot's current position and chose that centroid to move to it. This process repeats until all areas of the map are explored. Frontier edges are open cells adjacent to unknown cells, marked with an "x" below as shown in Figure (6) Frontier

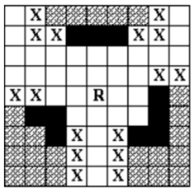

Fig. 6. fronter exploration cells

regions are groups of adjacent frontier regions, marked with different colors below. Each region's centroid is also marked as shown in Figure (7).

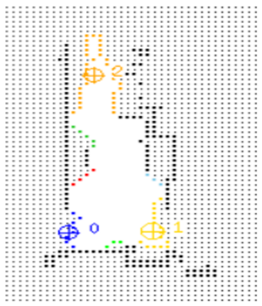

Fig. 7. fronter regions

- **Map Merging** Once all the maps were of the appropriate size and the robots we're able to explore frontiers, it was time to start merging the maps. Please see Figure (8) to understand the structure of merging multiple robot maps.

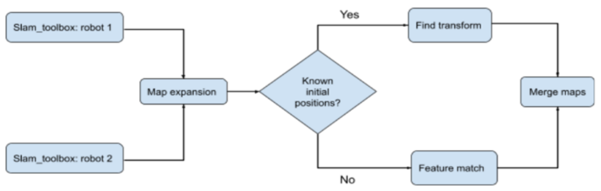

Fig. 8. the structure of merging multiple maps

For operations where you know the initial positions of the robots, map merging is relative straightforward. However, in a realistic setting, it's unlikely that someone would have that kind of control or knowledge. But without knowing the robot positions, how can we merge the maps? By stitching the sperate maps into one. Unfortunately, this functionality has a catch; you have to initially place the robots very close to each other so there is enough overlap for the feature-matching algorithm to work.
In my continued work on this project, I'll be tackling this issue and develop a method where the robots can start exploring without any information about the other while a feature detection algothrim works in the background to detected matching areas of the maps. Once common ground is found, we will then initiate the map merging process and reconfigure the robots to explore frontiers on the combined map instead of their local maps.

III. Proposed Robots Sensory System

The design of the robot shown in Figure (9).it should be simple and cost-effective. The robot must move autonomously and must have the necessary sensors to do so. The operating time of the robot must be long enough for the robot to do its tasks and achieve its missions properly and consistently. The operating time should be more than 1 hour. In our project we need to use suitable sensors to enable robot to map and navigate. Robot needs to know the position and the distance between itself and the obstacles.

Where is the robot? It is the first question we need to ask! the answer is "odometry". Sensors are mounted on the Robot to help achieve this mission of localizing the robot. There are two types of sensors: internal and external.

Internal sensors help monitor the kinetic and coordination state of the robot. Internal sensors used are:

1) compass
2) gyroscope
3) accelerometer
4) encoder

Beside having a full survey for the robot's speed, acceleration, and orientation, it is important for the robot to be aware of the surrounding environment. Thus, the robot is equipped with external sensors that can't only inform the robot with the surrounding obstacles but also determine the distance between the robot and these obstacles. External sensors like:

1) camera
2) ultrasonic sensor
3) LIDAR

- **LIDAR** Using the encoder is not enough, we also need to know what the world around the robot looks like. The LIDAR we use is RP LIDAR. it is a low cost laser rangefinder, 360 degree 2D laser scanner solution. The sensor can perform 360-degree scan within six meters' range. It scans by applying 5.5Hz when sampling 360 degrees each round. RP Lidar measures a distance based on a laser triangulation ranging principle and it uses high-speed vision acquisition. Mechanically, it emits modulated infrared laser signal and the laser signal is then reflected by the object to be detected. The returning signal

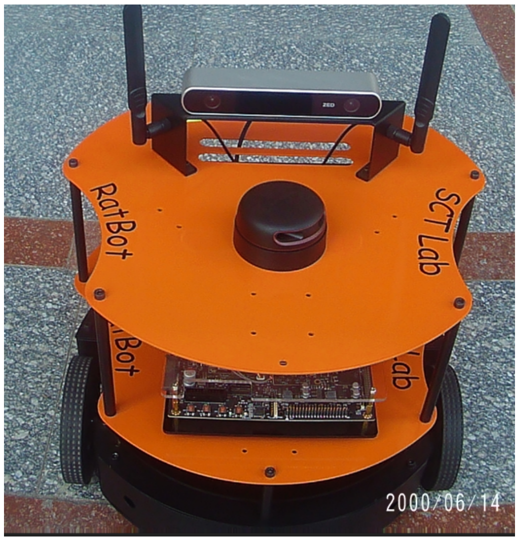

Fig. 9. ratbot

is sampled by a vision acquisition system in the RP Lidar. The RP LIDAR is shown in Figure (10)

Fig. 10. LIDAR

- **Encoder** An encoder is an electromechanical device which generates an electrical signal depending on the position or the displacement of the measured item. It is shown in Figure (11)In mobile robotics, rotary encoders are used to measure the movement (direction and speed) of each of the robot's wheels. Odometry is the use of motion sensors to determine the robot's change in position relative to some known position. For example, if a robot is traveling in a straight line and if it knows the diameter of its wheels, then by counting the number of wheel revolutions it can determine how far it has traveled. Robots will often have shaft encoders attached to their driving wheels which emit a fixed number of pulses per revolution. By counting these pulses, the processor can estimate the distance traveled. Based on the following assumptions and equations we can measure the linear distance that the robot moves.

$$D_c = \frac{D_l + D_r}{2} \tag{5}$$

$$x' = x + D_c \cos(\phi) \tag{6}$$

$$y' = y + D_c \cos(\phi) \tag{7}$$

$$\phi' = \phi + \frac{D_r - D_l}{L} \tag{8}$$

Assume each wheel has N "ticks "per revolution. Most wheel encoders give the total tick count since beginning. For both wheels:

$$\Delta tick = tick' - tick \tag{9}$$

$$D = 2\pi R \frac{\Delta tick}{N} \tag{10}$$

Fig. 11. encoder

IV. Experimental Results

The practical mission is to generate an indoor map for an alley inside a building, a map that the robot generates from type Occupancy Grid Map (OGM), which is commonly used in the ROS community. The map as shown in Figure (12) below, the white area is the free zone which the robot can move in, the black area is the occupied zone which the robot cannot move in, and the gray area is the unexplored.

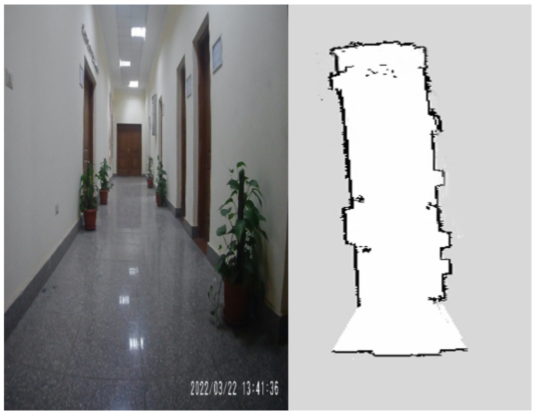

Fig. 12. Occupancy Grid Map (OGM)

- 0 is unoccupied (white).
- 100 is occupied (black).
- -1 is unexplored (gray).

The practical mission for the robotic swarm is making the swarm navigate an indoor environment using a map which is generated by another robot. we can apply navigation to robotic swarm as explained in the following steps 1.making network between three robots 2.open terminals into each robot through leader robot 2.opening ros in all robot from master robot 3.openning a map through RVIZ 4.give goal to leader robot 5.other robots generate goal as leader robot but with shifted.

When the occupancy probability is published as ROS message it can be extracted from the YAML file as an evenly spaced field of binary random variables each representing the presence of an obstacle at that location in the environment, where

After mapping, the second step is navigation. There are four steps to perform navigation.

- A navigation goal is sent to the nav stack to specify a goal pose (position and orientation) in some coordinate frame (commonly the map frame).
- The nav stack uses a path-planning algorithm in the global planner to plan the shortest path from the current location to the goal.
- This path is passed to the local planner, which tries to drive the robot along the path. The local planner uses information from the kinect in order to avoid obstacles that appear before the robot but that do not exist in the map, such as people. If the local planner gets stuck and cannot achieve progress, it can ask the global planner to make a new plan and then attempt to follow that.
- When the robot gets close to the goal pose, the action terminates and we're done.

The robot makes Autonomous navigation to go to goal by moving on path planning which makes as a shortest path and to avoid all obstacles as shown in Figure (13).

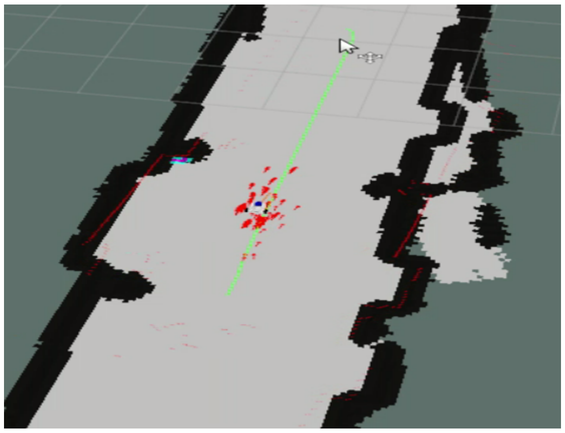

Fig. 13. path planning

V. Conclusions and Future Work

This paper presents a robot navigation and mapping for an indoor environment. the mapping method used generated a very accurate map compared to the environment. Single robot can navigate through this map and draw path which it moved on as the shortest pass and can avoid obstacles. robotic swarm is navigating in indoor environment by using the map which the single robot generated. give robot0 goal and the rest of the robots will take the same goal but with shift to move in an arrow head formation as shown in Figure (14).

In future, we hope to use all agent to map. The map is divided into sub-maps which are assigned for each robot to generate global map.

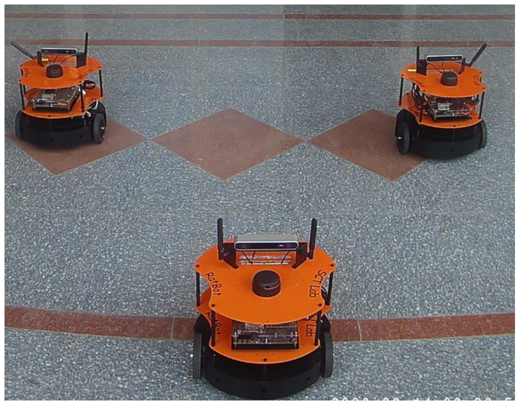

Fig. 14. robotic swarm pattern

References

[1] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," Scholarpedia, vol. 9, no. 1, p. 1463, 2014.

[2] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," Journal of Field Robotics, vol. 33, no. 1, pp. 3–46, 2016.

[3] D. Yi and Y. Guozheng, "Collaborative localization of multi micro-robots based on markov algorithm," in MHS2003. Proceedings of 2003 International Symposium on Micromechatronics and Human Science (IEEE Cat. No. 03TH8717). IEEE, 2003, pp. 349–355.

[4] V. A. Rosas-Cervantes, Q.-D. Hoang, S.-G. Lee, and J.-H. Choi, "Multi-robot 2.5 d localization and mapping using a monte carlo algorithm on a multi-level surface," Sensors, vol. 21, no. 13, p. 4588, 2021.