2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)

Development of a Miniature Robot for Multi-robot Occupancy Grid Mapping

Jugesh Sundram, Van Duong Nguyen, Gim Song Soh, Roland Bouffanais, Kristin Wood
Engineering Product Development
Singapore University of Technology and Design
Singapore
Email: sohgimsong@sutd.edu.org

***Abstract*—In this paper, we describe the design and evaluate the performance of Orion, a miniature sized differential drive robot for performing Intelligent Surveillance and Reconnaissance (ISR) tasks. The robot has a footprint of 12 x 12 cm and weighs a total of 150 grams. It is able to navigate over two dimensional surfaces and perform indoor environment sensing using a custom build laser based ranging module. It uses log-odd occupancy and an inverse sensor model for occupancy grid map construction. We show that with our developed system, the robot is able to perform mapping in a closed environment. We also show how multiple of such robots can be extended to perform multi-robot mapping. This is our first step in the development of a multi-robot system for indoor space mapping using an embedded system architecture.**

I. INTRODUCTION

The use of robots for the purpose of autonomous Intelligent Surveillance and Reconnaissance (ISR) is gaining prominence. The tasks include mapping of objective area, monitoring, detecting targets, tracking, or even search and rescue operations - with each task requiring robots to have a certain set of capabilities. Typically, these tasks are achieved through the use of Unmanned Aerial Vehicles (UAVs) or Unmanned Ground Vehicles (UGVs) depending on the type of terrain and mission requirements. However, these robots can prove bulky for any practical use as soldier systems thus making the use of a collaborative swarm of micro or miniature robots attractive.

Compared to UAVs, UGVs have the advantage of accurately locating ground targets [1] and capable of navigating into an unknown environment over an extended period of time. This makes them highly suitable for indoor mapping tasks. Various developments of miniature UGVs for autonomous ISR, each with its unique capabilities, can be found in the literature. [2]–[5]. However, these robots are limited to a demonstration of locomotion innovation. There is not much information available on the ability to perform mapping tasks of such robots. This could be due to the complexity and significant engineering efforts involved in implementing them onto an embedded system. The type of reliable sensors technology available that can fit within the footprint of such miniature UGVs is also important in order to make the system practical for soldiers to use.

Constructing a representative world model requires the robot to maintain good localization as it traverses into an unknown environment. This is challenging due to the localization nature in a GNSS denied environment where the uncertainty of the robot grows over time as it explore the unknown enviroment. The common world model for indoor mapping includes Occupancy Grid Map [6], [7], Line Maps [8], Topological Maps [9] and Landmark-Based Maps [10], with each having its own advantages and disadvantages. In our case, we are interested in using Occupancy Grid Maps as they offer metric based representation of the static environment. Since they use probabilistic estimations, they are tolerant towards moving objects in the environment that may not be a part of the map of the indoor space. They provide a clear representation of the known and unknown areas of the environment which can be further used for any adaptive path planning to explore the unknown environment.

There have been various initiatives towards designing intelligent surveillance systems for indoor environments. The robots discussed in [11], [12] have large size and weight footprints. The ranging module on board itself weighs in the order of kilos. The robot discussed in [13] is a vision-based surveillance robot that has 3 ultrasonic distance sensors to measure distance of obstacles from robot. Although ultrasonic sensors have smaller size and weight footprints, they may have several disadvantages in comparision to optical sensors such as the LiDAR ranging module presented in this paper. Ultrasonic sensors are intrinsically less accurate because sound is more difficult to focus than laser light. Accuracy is typically several centimeters and significantly higher in order of magnitude compared to the few millimeters accuracy of laser sensors. The accuracy directly affects the mapping results.

In this paper, we seek to display the ability of our miniature robot to perform multi-robot indoor space mapping. Our mapping system fuses the information obtained from the robot localizer and our custom built LiDAR ranging module to reconstruct a point cloud of the environment. This information is used to compute a representation of the environment through an Occupancy Grid Map [14]. The preliminary results presented here shows promising results and are the first step towards the development of a multi-robot robotic system for indoor space mapping.

II. THE MINIATURE ROBOT: ORION

The developed miniature robot, Orion in Fig. 1, employs a differential drive locomotion principle with a custom designed

978-1-5386-7066-8/18/$31.00 ©2018 IEEE

ranging module and a swarm enabling unit.

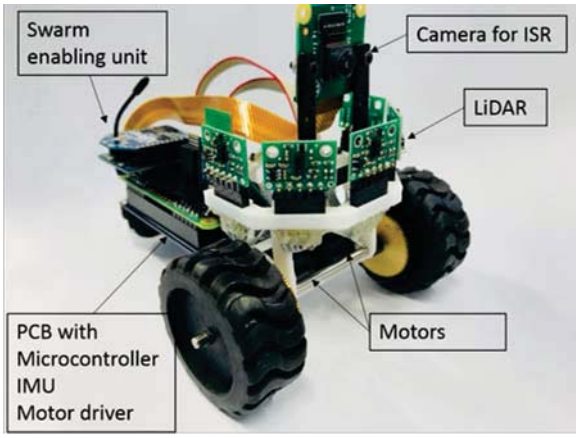

Fig. 1. Our miniature robot Orion with the designed LiDAR module static sensor array

The IMU onboard the robot is a Bosch Sensortec chip (BNO055). It belongs to a family of Application Specific Sensor Nodes (ASSN) implementing an intelligent Absolute Orientation Sensor, which includes the sensors and an on-board sensor fusion algorithm. We utilize its heading data (θ) for navigation and mapping purposes. A moving average bandpass filter is implemented to get an accurate estimate for the heading. The robot consists of 2 DC motors attached with encoders. The motors are controlled by a Toshiba motor controller (TB6612FNG). Pulsed data from the encoders is used to compute the incremental linear translation of the robot along the x and y axis. The Light Detection and Ranging (LiDAR) sensor module returns the point cloud data of the environment. All these sensors and controllers are interfaced with a STM32F411 microcontroller unit (MCU). The MCU chip collects data from sensors and sends it to a embedded computer in the Swarm Enabling Unit. We use the RaspberryPi to perform mapping. The RaspberryPi can also be used to interface with other robots through a ZigBee communication module and perform decentralized swarming [15].

A. Ranging Module

Conventional LiDAR systems consist of a single statically placed Laser emitting and sensing unit that scans across a certain Field of View (FoV) via shooting Laser beams through a rotating mirror that spans the FoV. They are able to measure long distances, in the order of 10 - 20 m, however they exceeded our specifications of size, weight and voltage consumption for a miniature robot.

To overcome this, we custom design our own ranging module using VL53L0X sensors. They are chosen due to their capability to scan a variety of indoor surfaces, small footprint, weigh and voltage consumption. The VL53L0X ranging sensor emits a 940nm Vertical Cavity Surface-Emitting Laser (VSCEL) and measures well upto $0.8m$. The sensors are also coupled with internal physical infrared filters that enable stable ranging readings and higher immunity to ambient light. This makes the sensors ideal for indoor use. This suited our needs as we are interested in mapping indoor environments using the robot. Therefore, Orion ranging module is designed using five statically placed VL53L0X Time-of-Flight ranging sensors as shown in Fig. 1.

III. MAPPING APPROACH

Our mapping approach fuses the information obtained from the robot localizer and the ranging module to construct the point cloud for use in building the occupancy grid map of the environment. The method called mapping with known poses. Currently, the localization of Orion relies on the measurements of the IMU and wheel encoder readings. Future work will involve localizing Orion by incorporating probabilistic localization algorithms such as Monte Carlo Localization [16] or Rao-Blackwellized Particle Filters [17], [18].

A. Point Cloud Mapping

To construct the point cloud, we made use of the five VL53L0X sensors, which are fixed at an interval of $\pi/4$ as shown in Fig. 2. Readings from each sensor is measured in

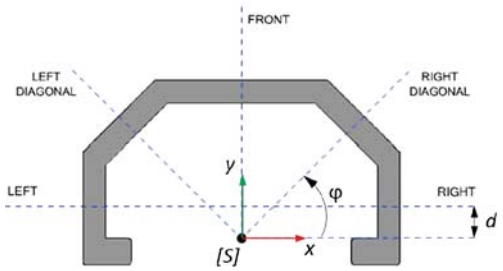

Fig. 2. The geometry of the LiDAR module static sensor array

the polar coordinate system $({}^S\rho_i, {}^S\varphi_i)$.

For each sensor i, the range reading is denoted by ${}^S\rho_i$ and is associated with an angle ${}^S\varphi_i$ as shown in Table I.

TABLE I
LIST OF ANGLE SUBTENDED BY EACH SENSOR

Sensor	φ_{index}	Angle (rad)
Right	φ_1	0
Right Diagonal	φ_2	$\pi/4$
Front	φ_3	$\pi/2$
Left Diagonal	φ_4	$3\pi4$
Left	φ_5	π

The reading in the polar coordinate is transformed to cartesian coordinate (x, y) using:

$$\begin{aligned} x_i &= {}^S\rho_i \cos({}^S\varphi_i) \\ y_i &= {}^S\rho_i \sin({}^S\varphi_i) \end{aligned} \tag{1}$$

The coordinate (x_i, y_i) here represents the point cloud data ${}^SP$ of the environment with respect to the sensor's local frame of reference $[S]$. If the sensor is placed at a displacement ${}^R[T]_S$ with respect to the robot's frame of reference $[R]$, then:

$${}^R[T]_S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & d_{RS} \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where d_{RS} is the linear translation between $[S]$ and $[R]$.

In order to represent the point cloud data ${}^{S}P$ with respect to a fixed world reference frame $[W]$, we use the following coordinate transformation:

$$ {}^{W}P = {}^{W}[T]_{R}\,{}^{R}[T]_{S}\,{}^{S}P \tag{3} $$

where

$$ {}^{W}[T]_{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \tag{4} $$

denotes the displacement of the robot relative to the fixed world frame [W].

B. Occupancy Grid Mapping

The point cloud data obtained from Eq. (3) needs to be stitched together to represent a map of the environment. We use a discrete grid of cells called the Occupancy Grid Map to represent the map of the environment. Each cell in this discrete grid is assigned a posterior probability of occupancy based on the pose $\mathbf{x}_t$ at time t and point cloud data ${}^{W}P$ at time t represented as $\mathbf{z}_t$.

We assume that a grid m is a matrix with certain number of rows r and columns c where each cell is represented as m_i as shown in Fig.3. The posterior probability estimate of occupancy of each cell m_i given the sequence of all the poses $\mathbf{x}_{1:t}$ until time t and set of all point cloud data $\mathbf{z}_{1:t}$ until time t is represented as follows:

$$ p(m_i = occupied | \mathbf{z}_{1:t}, \mathbf{x}_{1:t}) \tag{5} $$

We define a term $l_{t,i} \in (-\infty, \infty)$ as the log-odds occupancy at time t at cell index i. It is a value that accumulates based on the occupancy of a cell. An occupied cell has a high $l_{t,i} = l_{occ}$ value and conversely, a free cell has a low $l_{t,i} = l_{free}$ value. This value is typically computed from a sensor model of the ranging sensors using

$$ l_{t,i} = \log \frac{p(m_i | \mathbf{z}_{1:t}, \mathbf{x}_{1:t})}{1 - p(m_i | \mathbf{z}_{1:t}, \mathbf{x}_{1:t})} \tag{6} $$

but we find that good results can still be obtained by setting l_{occ} and l_{free} empirically without using Eq. (6). Thus, for our case, the posterior probabilities are recovered from $l_{t,i}$ in the following way:

$$ p(m_i | \mathbf{z}_{1:t}, \mathbf{x}_{1:t}) = 1 - \frac{1}{1 - \exp\{l_{t,i}\}} \tag{7} $$

Note that, the main advantage of using the log-odds occupancy $l_{t,i}$ over directly using the posterior probability to compute occupancy is that numerical instabilities for probabilities near zero or one are avoided.

The `OccupancyGridMapping` function as shown in Algorithm 1 loops through all grid cells i and updates the cells that are within the perceptual Field-of-View of the sensor with values returned by the `InvSensor` function as shown in Algorithm 2.

A graphical display of the `OccupancyGridMapping` procedure is shown in Fig.3. Let us consider an $r \times c$ occupancy grid map m. Each cell in the grid m_i is a posterior probability estimate of occupancy that is computed by assigning a log-odds occupancy value $l_{t,i}$ at a given time t. As such, the prior of occupancy represented as log-odds occupancy is $l_{prior} = 0$ for all the cells in m since there is no information whether a cell is occupied or free.

Algorithm 1 Occupancy Grid Mapping

procedure OCCUPANCYGRIDMAPPING($\{l_{t-1,i}\}, \mathbf{x}_t, \mathbf{z}_t$)
  for all cells m_i **do**
    if m_i is in the perceptual FoV of the sensor **then**
      $l_{t,i} \leftarrow l_{t-1,i} + \texttt{InvSensor}(m_i, \mathbf{x}_t, \mathbf{z}_t) - l_0$
    else
      $l_{t,i} \leftarrow l_{t-1,i}$
    end if
  end for
  return $\{l_{t,i}\}$
end procedure

Let us assume that the real world is walled on the north and west side. These would correspond to the first row $(m_1, m_2 \ldots m_6)$ and first column $(m_1, m_7 \ldots m_{31})$ on the occupancy grid map. We will run a single Orion robot in this environment across a time space of 3 time steps. The robot begins from cell m_{36} at $t = 1$ and move to cell m_{35} at $t = 2$ and to cell m_{34} at $t = 3$. To show the effectiveness of occupancy grid mapping, we will assume that at $t = 1$ there is exists a dynamic obstacle at m_{10}. It vanishes from m_{10} for $t = 2, 3$.

By knowing the pose $\mathbf{x}_t$ and the point cloud measurement $\mathbf{z}_t$, it is possible to determine the cells that are within the perceptual Field-of-View of the sensor. This is done by first determining the cells that will correspond to $\mathbf{x}_t$ and $\mathbf{z}_t$ in the occupancy grid map m. This can be computed by a function called `GRID`.

Let's say the m_i corresponding to $\mathbf{x}_t$ is denoted by x and the m_i corresponding to $\mathbf{z}_t$ is denoted by z. As shown in Fig.3(a), let us consider the robot position x at m_{36} and the `RIGHT` sensor measurement z of the LiDAR module hitting the cell at m_{10}. A ray casting algorithm such as the Bresenham algorithm [19] is used to find the cells that form a ray between x and z. In our case, the cells are m_{29}, m_{23}, m_{17} and m_{16}. These cells are within the perceptual Field-of-View of the `RIGHT` sensor.

At $t = 2$, the robot moves to cell m_{35}, the dynamic obstacle at m_{10} disappears rendering m_{10} to be a free cell. The value of $l_{2,10}$ lowers and the $p(m_{10} = occupied)$ moves away from 1. The `RIGHT` sensor detects the wall at m_4 thus $l_{2,4}$ is set at a positive value making $p(m_{10} = occupied) \rightarrow 1$. Meanwhile, the wall at m_{19} is detected by the `RIGHT DIAGONAL` sensor.

At $t = 3$, robot moves to m_{34}, the `RIGHT` sensor detects the wall again at m_4 thus $l_{3,4}$ accumulates a higher positive value. At cell m_{10}, the value $l_{3,4}$ further depreciates and thus the $p(m_{10} = occupied) \rightarrow 0$. Meanwhile, the wall at m_{13} and m_{31} are detected by the `RIGHT DIAGONAL` and `FRONT` sensors respectively.

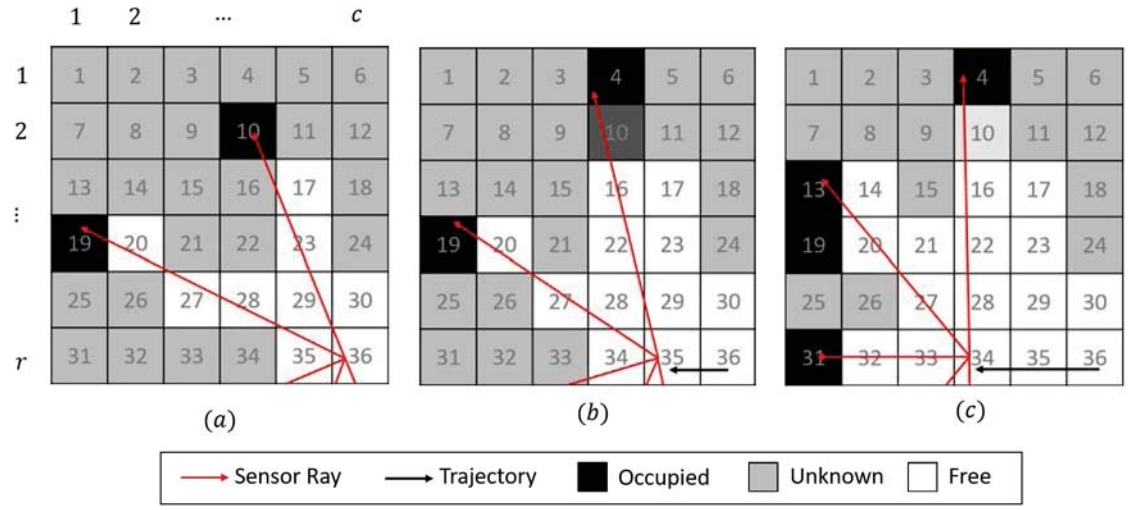

Fig. 3. Occupancy Grid Map consisting of r rows and c columns: (a) At $t = 1$, robot starts at m_{36} detects a dynamic obstacle at m_{10} and a wall at m_{19}. (b) At $t = 2$, robot moves to m_{35}, dynamic obstacle at m_{10} disappears making m_{10} a free cell, wall at m_4 is detected. (c) At $t = 3$, robot moves to m_{34}, probability of occupancy at m_{10} inclines towards representing a free cell, wall at m_{13} and m_{31} detected.

C. Inverse Sensor Model

The Inverse Sensor Model `InvSensor` as shown in Algorithm 2 assigns a log-odds value of occupancy for cells that are within the perceptual Field-of-View of the sensor. For cells that are classified as occupied, the l_{occ} is set a high positive value and for the cells classified as free the l_{free} is set at a significantly lower value than l_{occ}, ideally a negative number.

Algorithm 2 Inverse Sensor Model

procedure INVSENSOR($m_i, \mathbf{x}_t, \mathbf{z}_t$)
  $z \leftarrow ManhattanDistance(\texttt{GRID}(\mathbf{x}_t) - \texttt{GRID}(\mathbf{z}_t))$
  $r \leftarrow ManhattanDistance(\texttt{GRID}(\mathbf{x}_t) - m_i)$
  if $|r - z| < 1$ **then**
    return $\{l_{occ}\}$ ▷ l_{occ} is set a high positive value
  end if
  if $r < z$ **then**
    return $\{l_{free}\}$ ▷ l_{free} is set a negative value
  end if
end procedure

D. Multi-Robot Mapping

The true power of swarm robotics lies in the ability to breakdown a task such that it is performed by multiple systems for a fraction of cost and resources spent by a single system to perform the task on its own.

Given our task to map indoor environments, we can leverage on the mapping abilities of multiple Orion robots. Each robot will store a map of the environment it traverses. Knowing the initial start positions of these robots, the maps stored in each robot can be assimilated and aligned to create a global map. Knowing the start positions of the robots gives us information about the relative pose between robots. This relative pose information between each robot can be used to perform planar transformations of the point cloud data and trajectory data from each robot's initial position frame of reference to a common frame of reference. All the data in this common frame of reference can now be used to construct an Occupancy Grid Map using the `OccupancyGridMapping` algorithm.

IV. EXPERIMENT

In the following, we will now show the mapping results of our Orion robot.

A. Single Robot Run

The Orion robot was made to run inside a rectangular shaped space as shown in Fig. 4. Multiple waypoints were assigned inside this space such that the robot finishes a loop and maps the space. Raw sensor value was obtained from the LiDAR module. The point cloud data output of the LiDAR module was processed offline and the result was plotted and overlayed on the rectangular shaped space as shown in Fig. 5.

The point cloud data measurements from the LiDAR module was used to compute an occupancy grid map and the result is as shown in Fig. 6. The grey colored cells in the occupancy grid represent unknown regions of the map and have a prior $p(m_i = occupied) = 0.5$. The white colored cells represent free regions and the darker colored cells represent occupied regions. This map can be further improvised by doing multiple runs of the same space using different waypoints. A significant advantage of using occupancy grid maps is they are great in ignoring dynamic obstacles which may not be permanent features of the map.

B. Multi-Robot Run

To display the ability of Orion to perform multi-robot mapping, 2 Orion robots, Robot 1 and Robot 2, were made to run inside two distinct spaces, Space 1 and Space 2, as shown

Fig. 4. Rectangular space

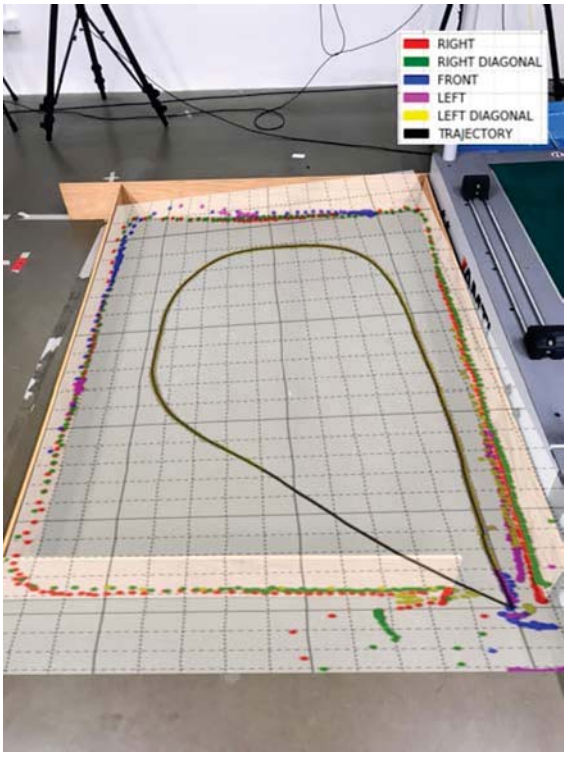

Fig. 5. LiDAR output with trajectory of robot overlayed on rectangular space

in Fig. 7. Space 1 consisted of varied shaped walls and Space 2 consisted of a rectangular shaped space. Robot 1 was made to run inside Space 1 and Robot 2 was made to run insider Space 2. Similar to the single robot run experiment, multiple waypoints were inside the spaces such that the robots finish a loop and map the spaces.

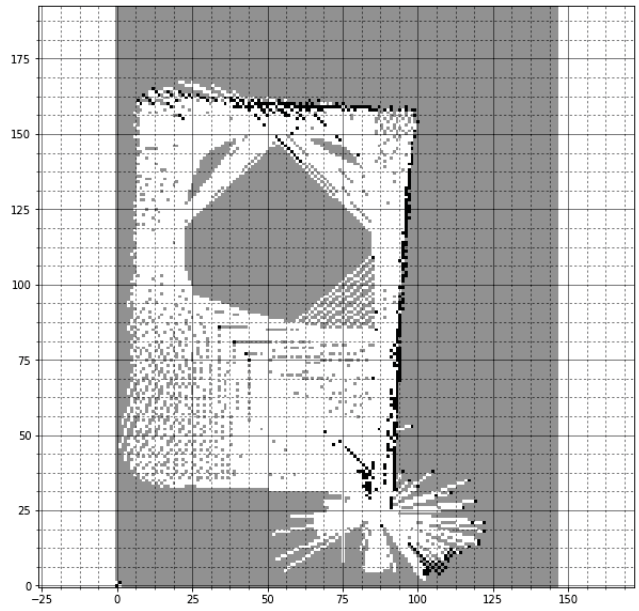

Fig. 6. Occupancy Grid Map

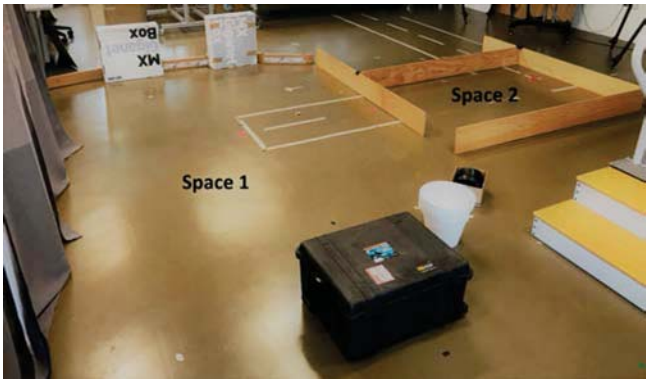

Fig. 7. Multiple Space Environment

We may consider the common reference frame $[C]$ coinciding with the frame of reference at the initial position of Robot 1 $[R1]$. The initial position of Robot 2 $[R2]$ was measured to be at a position $(2150\,\mathrm{mm}, 1450\,\mathrm{mm}, \frac{14\pi}{9}\mathrm{rad})$ relative to $[R1]$. The point cloud data and trajectory data of Robot 2 relative to $[R2]$ is transformed to $[C]$ using ${}^{C}[T]_{R2}$ as described in Eq. (8). The result of this transformation is shown in Fig. 8.

$$ {}^{C}[T]_{R2} = \begin{bmatrix} \cos(14\pi/9) & -\sin(14\pi/9) & 2150 \\ \sin(14\pi/9) & \cos(14\pi/9) & 1450 \\ 0 & 0 & 1 \end{bmatrix} \quad (8) $$

The transformed point cloud and trajectory value from both the robots is now used to construct an Occupancy Grid Map as shown in Fig. 9. This approach can be extended for any number of robots.

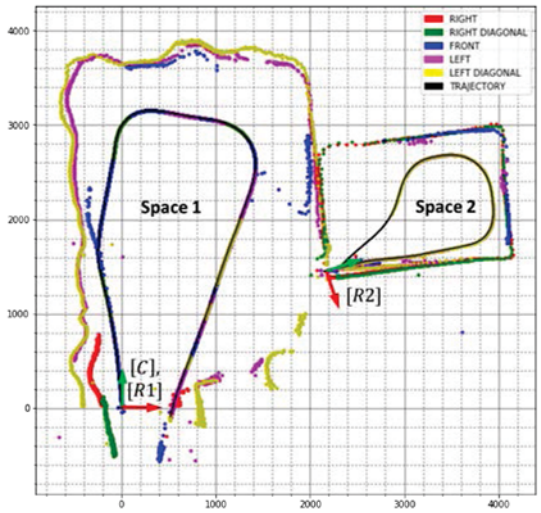

Fig. 8. Data from the two robots transformed to a common reference frame $[C]$

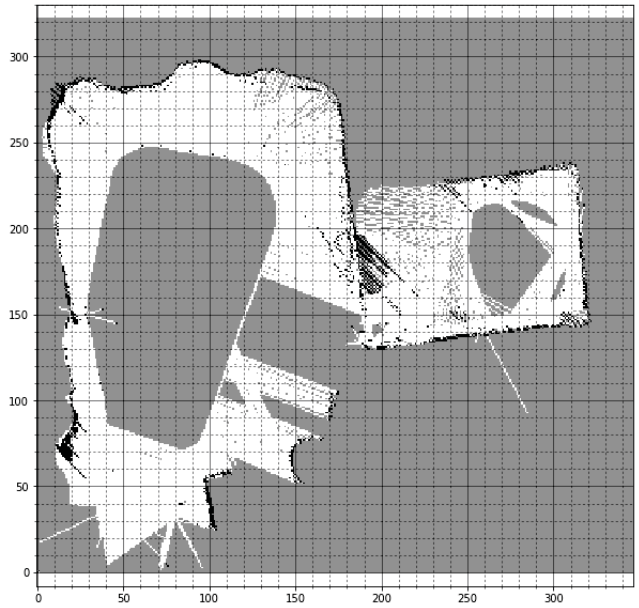

Fig. 9. Occupancy Grid Map of multiple spaces stitched into a common map

V. Conclusion

This paper presented Orion, a miniature robot with an embedded system architecture that is able to perform mapping in closed indoor environments. Multiple such robots can be made to operate in a swarm to perform multi-robot mapping of indoor environments and construct an occupancy grid map. Some future objectives include running the robot autonomously and performing Simultaneous Localization and Mapping. These are currently under development and will be presented at a later stage.

Acknowledgment

The authors would like to thank Temasek Laboratories@SUTD, Singapore for their financial support and HopeTechnik, Singapore for their technical assistance at various stages of the project.

References

[1] B. Grocholsky, J. Keller, V. Kumar & G. Pappas (2006). Cooperative air and ground surveillance. *IEEE Robotics & Automation Magazine, 13*(3), 16-25.

[2] Satria, S., Foong, S., Soh, G. S., & Wood, K. L. (2017). Robust Variable Phase Shaper for vibration suppression of start-stop motion of a spherical rolling robot. *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)* (pp. 1675-1681).

[3] Rybski, P. E., et al. (2000). Enlisting rangers and scouts for reconnaissance and surveillance. *IEEE Robotics & Automation Magazine, 7*(4), 14-24.

[4] Dan, Z., Tianmiao, W., Jianhong, L., & Guang, H. (2004). Modularization of miniature tracked reconnaissance robot. *IEEE International Conference on Robotics and Biomimetics (ROBIO)* (pp. 490-494).

[5] Xiao, J., Morris, W., Chakravarthy, N., & Calle, A. (2006). City climber: a new generation of mobile robot with wall-climbing capability. *Unmanned Systems Technology VIII* (pp. 62301D).

[6] A. Elfes. (1989). Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer, 22*(6), 46-57.

[7] Korthals, T., Barther, M., Schpping, T., Herbrechtsmeier, S., & Rckert, U. (2016). Occupancy grid mapping with highly uncertain range sensors based on inverse particle filters. In ICINCO 2016 - Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics (Vol. 2).

[8] D.H. Douglas, & T.K. Peucker. (1973). Algorithms for the reduction of the number of points required to represent a line or its caricature, *Cdn. Cartogr. 10*(2), 112?122.

[9] H. Choset, & K. Nagatani. (2001). Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization, *IEEE Trans. Robot. Autom. 17*(2), 125?137.

[10] M. Montemerlo, S. Thrun, D. Koller, & B. Wegbreit. (2002). FastSLAM: a factored solution to the simultaneous localization and mapping problem, *Proc. Nat. Conf. Artif. Intell. (AAAI)*.

[11] D. Di Paola, A. Milella,G. Cicirelli, A. Distante (2010). An autonomous mobile robotic system for surveillance of indoor environments. *International Journal of Advanced Robotic Systems 7*,(1), (pp. 8).

[12] D.A. Carnegie, A. Prakash, C. Chitty, B. Guy (2004). A human-like semi autonomous mobile security robot. *Internation Conference on Autonomous Robots Agents* (pp. 64-69).

[13] W. Budiharto (2015). Intelligent surveillance robot with obstacle avoidance capabilities using neural network. *Computational intelligence and neuroscience* (p.52).

[14] S. Thrun, W. Burgard and D. Fox (2005). Probabilistic robotics. MIT Press.

[15] Chamanbaz, M., Mateo, D., Zoss, B. M., Tokić, G., Wilhelm, E., Bouffanais, R., & Yue, D. K. (2017). Swarm-enabling technology for multi-robot systems. *Frontiers in Robotics and AI, 4*, 12.

[16] S. Thrun,D. Fox, W. Burgard, F. Dellaert (2001). Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2), (pp.99-141).

[17] A. Doucet, J. de Freitas, K. Murphy, S. Russel (2000). Rao-Blackwellized particle filtering for dynamic Bayesian networks. in *Proc. Conf. Uncertainity Artificial Intelligence*, Stanford, CA, 2000 (pp. 176 - 183).

[18] K. Murphy (2000), Bayesian map learning in dynamic environments. in *Proc. Conf. Neural Inf. Process. Systs.*, Denver, CO, 1999 (pp. 1015-1021).

[19] J.E. Bresenham (1965). Algorithm for computer control of a digital plotter. *IBM Systems Journal* 4 (No. 1), (pp. 25-30)