Rao-Blackwellized Particle Filter with Neural Network using Low-cost Range Sensor in Indoor Environment

Norhidayah Mohamad Yatim[1], Amirul Jamaludin[2], Zarina Mohd Noh[3]
Centre for Telecommunication Resesarch & Innovation (CeTRI),
Fakulti Kejuruteraan Elektronik dan Kejuruteraan Komputer (FKEKK)
Universiti Teknikal Malaysia Melaka (UTeM)
76100 Melaka, Malaysia

Norlida Buniyamin[4]
Faculty of Electrical Engineering (FKE)
Universiti Teknologi MARA (UiTM)
Shah Alam, 40450
Selangor, Malaysia

***Abstract*—Implementation of Rao-blackwellized Particle Filter (RBPF) in grid-based Simultaneous Localization And Mapping (SLAM) algorithm with range sensors commonly developed by using sensor with dense measurements such as laser rangefinder. In this paper, a more cost convenient solution was explored where implementation of array of infrared sensors equipped on a mobile robot platform was used. The observation from array of infrared sensors are noisy and sparse. This adds more uncertainty in the implementation of SLAM algorithm. To compensate for the high uncertainties from robot's observations, neural network was integrated with the grid-based SLAM algorithm. The result shows that the grid-based SLAM algorithm with neural network has better accuracy compared to the grid-based SLAM algorithm without neural network for the aforementioned mobile robot implementation. The algorithm improves the map accuracy by 21% and reduce robot's state estimate error significantly. The better performance is due to the improvement in accuracy of grid cells' occupancy value. This affects the importance weight computation in RBPF algorithm hence resulting a better map accuracy and robots state estimate. This finding shows that a promising grid-based SLAM algorithm can be obtained by using merely array of infrared sensors as robot's observation.**

Keywords*—Simultaneous Localization And Mapping (SLAM); occupancy grid map; Neural Network; Rao-Blackwellized Particle Filter; infrared sensor

I. Introduction

Sensor is an important element in mobile robot. Sensor such as vision sensor, range sensor and depth sensor are commonly serves as robot's observation of the world. Depending on the environment and task of the robot, different sensor have different advantages. For example, for underwater applications, acoustic based sensor such as sonar sensor is widely applied [1] while for drone navigation, vision sensor is more suitable [2]. Similar to other fields, a cost-effective solution is also a fundamental requirement in mobile robot implementation. Although, current sensors are very reliable for autonomous navigation, but to implement these fast and high accuracy sensor comes with significant cost. Hence, there is research that focuses on developing low-cost navigational method [3]. To achieve this objective, sparse and noisy sensors such as ultrasonic sensor or infrared sensor modules is an alternative to reduce the overall cost [4], [5]. Infrared sensor is commonly used in array configuration on robot's platform to avoid obstacles using obstacles avoidance algorithm such as Braitenberg algorithm [6]. This configuration gives sparse measurements as the measurements only available at certain angles. In contrast to range sensor such as laser ranger finder or ratating infrared sensor that gives more dense measurements at much higher cost.

There are previous works in robot's simultaneous localization and mapping or SLAM that uses sparse and noisy sensors with neural network as a method to evaluate cells' occupancy in grid map [7], [8], [9]. Using neural network to interpret sensor measurements into occupancy grid cell gives multiple advantages on noisy and sparse sensors implementation. The first one is that from the training data, neural network can generally distinguish erroneous sensor measurements or also called "maximum range" readings. These readings can be caused by poor reflecting surfaces such as glass materials [10] and in situations where no obstacle is within the measurement range of the sensor. Neural network is a simpler approach compared to identify the properties of surface's reflection and compute angle of incidence of sensor's beam to differentiate erroneous measurements [11]. Another advantage is that, the noise caused by injective and non-linear sensor model of infrared sensor can be reduced based on training data [9]. Other than that, neural network can interpret multiple sensors reading concurrently rather than interpreting range measurements independently. This way, the correlation between adjacent sensors can be exploited. This can gain more information for sensors in ring or array formation [7], [12].

To implement grid-based SLAM with array of infrared sensors, the proposed method is to counterbalance the limitation of sparse measurement produce from array of infrared sensor by expanding or extrapolating the sensor measurements. Thus, in this research, the integration of neural network with grid-based SLAM algorithm were developed and experimented with array of infrared sensors.

Overall in this paper, Section 2 describes the method used to carry out the experiments, which consists of the framework for Rao-blackwellized Particle Filter algorithm and neural network configuration. Next, Section 3 reports the results of neural network training, robot's state estimate and map estimate of the SLAM algorithm described. The analysis of the algorithm performance is included as well. Lastly, Section

4 concludes the results of this paper.

II. METHOD

In this section, the Rao-blackwellized Particle Filter (RBPF) algorithm with neural network are explained and elaborated. This consists of the method applied in each component of the RBPF framework.

A. Rao-Blackwellized Particle Filter Algorithm

The Rao-Blackwellized Particle Filter uses particles to estimate robot's state, like the particle filter algorithm. Here, each particle maintains a hypothesis of robot pose that assume its position is correct. Each particle build a map based on its trajectory which is also their own version of the map [13]. The algorithm of learning grid maps using RBPF or grid-based SLAM algorithm is described as follows:

1) **Sampling**: Sample i^{th} particle, $x_t^{[i]}$ to generate current set of particles, X_t by sampling from a proposal distribution.
2) **Importance weighting**: Calculate the importance weight of i^{th} particle, $w_t^{[i]}$, which accounts for how the robot's observation, z_t match with the current map estimate, $m^{[i]}$.
3) **Map update**: Update the particles' map estimate, $m^{[i]}$, conditioned on robot's state, $x_t^{[i]}$ and robot's observation, z_t.
4) **Density extraction**: Calculate the density of X_t by using weighted mean method. The density of X_t, is the single state that represent the robot's position state estimate, x_{est}. The map update for x_{est} is computed as well.
5) **Resampling**: Resample the particles proportionally to the particles' weight. Particles with higher weight will be most likely to be resampled for the next generation of particles.

B. Sampling from Motion Model

In the sampling step, the current set of particles, X_t is sampled from a proposal distribution. In this paper, the particles, X_t is sampled from motion model proposal distribution to predict robot's state. Looking at the ideal case, if particles can be sampled from a true target distribution, the weight of all particles will be equal and the same set of particles, X_t can be maintained without the resampling step [14]. However, a closed form of this posterior is not available in general. As a result, typical method is to sample from motion model, $p(x_t|a_t, x_{t-1})$ as the proposal distribution and use the observation model, $p(z_t|m, x_t)$ as the importance weight.

The motion model makes used of robot's transition model which can be modelled either using velocity information or odometry measurements [15]. For this research, odometry-based model is used by utilizing the wheel encoders' measurements in the robot system. A transition model for two wheels robot from [16] is adapted.

Equation (1) describes the transition from previous state, $x_{t-1} = (x, y, \theta)^T$ to current state, $\bar{x}_t = (\bar{x}, \bar{y}, \bar{\theta})^T$. The notation $\bar{x}_t$ is used (instead of x_t) to denote that this is the predicted value of current robot's state. The robot movement is illustrated in Fig. 1. Here, l and r are the traveled distance by left wheel and right wheel respectively. While b is the distance between two wheels. $\Delta\theta$ is the change of robot orientation where the calculation is $\Delta\theta = (r - l)/b$ [16].

$$\begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{\theta} \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} \frac{r+l}{2}\cos(\theta + \frac{r-l}{2b}) \\ \frac{r+l}{2}\sin(\theta + \frac{r-l}{2b}) \\ \frac{r-l}{b} \end{pmatrix} \quad (1)$$

The essence of probabilistic SLAM is to model measurements by taking into account the uncertainty that comes with the measurements. The odometry measurements were described using probabilistic motion model $p(x_t|x_{t-1}, a_t)$. The motion model defines uncertainties or noise from transition model described earlier. $p(x_t \mid x_{t-1}, a_t)$ gives the likelihood of a robot current position, x_t given that previously it was at x_{t-1}. Subsequently, it moves and odometry data, a_t is measured. The odometry motion model makes used of robot's odometry data. Odometry motion model sampling is an approach where the relative odometry at time $t-1$ to t is divided into three consecutive actions; initial turn, δ_{rot1}, a translation, δ_{trans}, and a second turn, δ_{rot2} as illustrated in Fig. 2 [15].

Algorithm 1 shows the complete algorithm to sample from motion model distribution. The relative odometry is calculated using trigonometry (see line 4 to 6). At line 7 to 9, noise terms; ϵ_{rot1}, ϵ_{trans}, and ϵ_{rot2} are sampled from Gaussian probability distribution with mean zero and the standard deviation of robot-specific parameters, α_1 to α_4. These parameters specify the noise in robot motion where α_1 and α_4 are the rotational errors or angular error, and α_2 and α_3 are the translational errors. These errors come with the motion of relative odometry, δ_{rot1}, δ_{trans}, and δ_{rot2}. In the next subsection the computation of importance weight of each particle is explained.

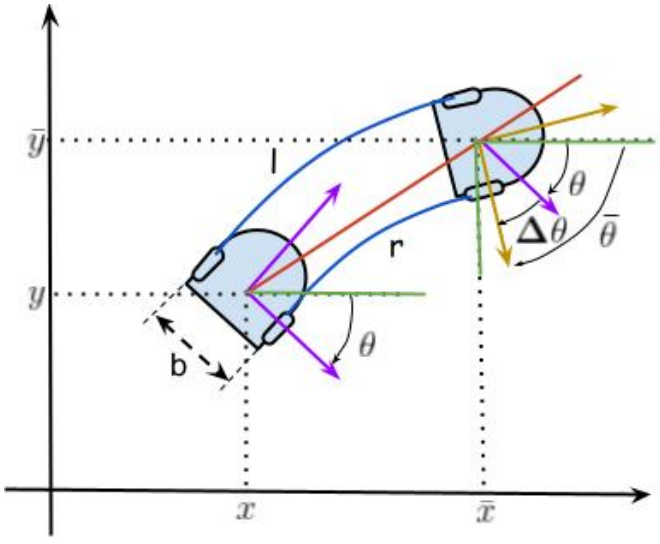

Fig. 1. Robot Moves from $(x, y, \theta)^T$ to $(\bar{x}, \bar{y}, \bar{\theta})^T$.

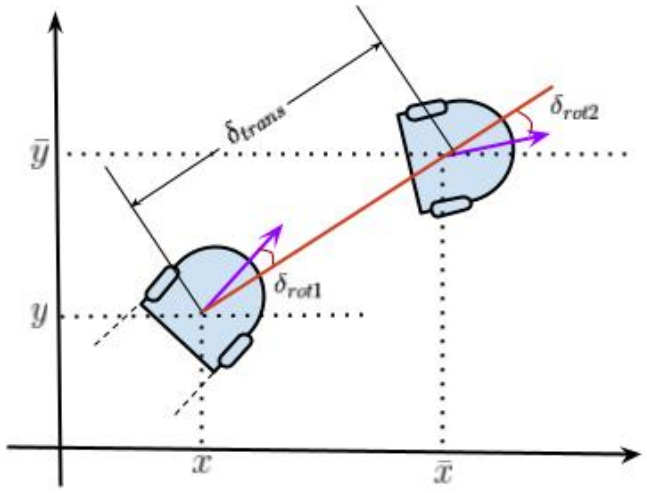

Fig. 2. Relative Odometry shows Initial Turn, δ_{rot1}, a Translation, δ_{trans}, and a Second Turn, δ_{rot2}.

Algorithm 1 Sampling with Odometry Motion Model

Require: x_{t-1}, a_t
1: $\bar{x}_t = g(x_{t-1}, a_t)$
2: $\langle x, y, \theta \rangle = x_{t-1}$
3: $\langle \bar{x}, \bar{y}, \bar{\theta} \rangle = \bar{x}_t$
4: $\delta_{rot1} = atan2(\bar{y} - y, \bar{x} - x) - \bar{\theta}$
5: $\delta_{trans} = \sqrt{(\bar{x} - x)^2 + (\bar{y} - y)^2}$
6: $\delta_{rot2} = \bar{\theta} - \theta - \delta_{rot1}$
7: $\epsilon_{rot1} \sim \mathcal{N}(0, \alpha_1 |\delta_{rot1}| + \alpha_2 \delta_{trans})$
8: $\epsilon_{trans} \sim \mathcal{N}(0, \alpha_3 \delta_{trans} + \alpha_4 |\delta_{rot1}| + \alpha_4 |\delta_{rot2}|)$
9: $\epsilon_{rot2} \sim \mathcal{N}(0, \alpha_1 |\delta_{rot2}| + \alpha_2 \delta_{trans})$
10: $\hat{\delta}_{rot1} = \delta_{rot1} + \epsilon_{rot1}$
11: $\hat{\delta}_{trans} = \delta_{trans} + \epsilon_{trans}$
12: $\hat{\delta}_{rot2} = \delta_{rot2} + \epsilon_{rot2}$
13: $x' = x + \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1})$
14: $y' = y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$
15: $\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$
16: **return** $x_t = (x', y', \theta')^T$

C. Importance Weight with Map Matching

Map-matching algorithm is used to obtain the importance weight of i^{th} particle, $w_t^{[i]}$ at current time step, t. The map matching method uses similarities of local and global map of a particle. In this research, the map matching approach uses a match function introduced in [17]. The function compares the local map and the global map of each i^{th} particle and recorded the number of match values. The match number, $match$ is then applied to the function in equation 2. In this equation, f is a parameter that will influence the distribution of particles' weight.

$$w_t^{[i]} = w_{t-1}^{[i]} e^{\frac{match^{[i]}}{f}} \tag{2}$$

The if and else cases in (3) shows how $match$ is calculated. A local map denotes by m_l and a global map denotes as m_g. $p(m_{g,j})$ and $p(m_{l,j})$ are the probability of occupancy for global map and local map at jth cell respectively. While occ and $free$ are the threshold values that a cell is considered occupied or free respectively. The $match$ value is added by one if the jth cell of both local and global map exceed the occ threshold value. While for a contradict value of jth cell, the $match$ value is deducted by one.

$$match = \begin{cases} +1, p(m_{g,j}) \geq occ \wedge p(m_{l,j}) \geq occ \\ -1, p(m_{g,j}) \geq occ \wedge p(m_{l,j}) \leq free \\ -1, p(m_{g,j}) \leq free \wedge p(m_{l,j}) \geq occ \end{cases} \tag{3}$$

D. Selective Resampling

After the computation of weight for each particle, particles are resampled proportionally to their weight. The resampling step is executed based on a heuristic measure of how well the proposal distribution approximates the target distribution called number of effective sample size, N_{eff}. N_{eff} measures the variance of particles' weights. The N_{eff} value is calculated using equation (4), which is the reciprocal of the sum of squares of all particles' weight. Here, N is the number of particles or sample size. Practically, the normalized value of particles weight is used in the N_{eff} calculation.

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w^i)^2} \tag{4}$$

A lower N_{eff} value reflects that there is high variance in particle's weight distribution. This signals that some of the particles have significantly higher weight compared to the others. Algorithm (2) shows integration of selective resampling in particle filter algorithm whereas particles only resampled when N_{eff} value falls below a certain threshold. In this research, the threshold is set to $N/2$ [18]. Thus, particles are only resampled if N_{eff} value falls below half of the number of particles. This shows that the variance of particles' weight is significantly large and particles need to be resampled in order not to waste the particles on particles with low weights. If N_{eff} value is still within the threshold, all the particles are kept, with their normalized weights (see line 1 to 3). Normalization step gives more significant weight values to all particles.

Algorithm 2 Selective Resampling Algorithm

Require: $w_t^{[i]}, i \in \{1, \ldots, N\}$
1: **while** $i \in N$ **do**
2: $\quad w_t^{[i]} = w_t^{[i]} \times [\sum_i^N w_T^{[i]}]^{-1}$
3: **end while**
4: $N_{eff} = [\sum_{i=1}^{N} (w^i)^2]^{-1}$
5: **if** $N_{eff} < threshold$ **then**
6: $\quad X_t = \{\}$
7: $\quad$ **while** $i \in N$ **do**
8: $\quad\quad$ draw $x_t^{[i]}$ with probability $\propto w_t^{[i]}$
9: $\quad\quad X_t = X_t \cup \{\langle x_t^{[i]}, 1 \rangle\}$
10: $\quad$ **end while**
11: **end if**
12: **return** X_t

E. Particle Density

As particle filter maintains a set of hypotheses of robot's state, the estimates of robot's state need to be represented. In other words, from the set of particles, X_t, the density of the particles, x_{est} is computed to evaluate the performance of the SLAM algorithm. There are multiple methods that can be implemented to estimate particles' density. One of the method is to calculate the weighted mean of all particles as the overall robot's state estimate, x_{est}. For the x and y state of x_{est}, equation (5) is used, where w^i is the weight of each particle. As for robot's heading, θ_{est}, a separate equation (6) is used to prevent discretization error in the mean heading.

$$x_{est} = \sum_{i=1}^{N} w^i x^i \tag{5}$$

$$\theta_{est} = \arctan \frac{\sum_{i=1}^{N} \cos(\theta^i) w^i}{\sum_{i=1}^{N} \sin(\theta^i) w^i} \tag{6}$$

F. Neural Network Configuration

In this research, neural network served as the inverse sensor model, $p(m_t|x_t, z_t)$ to evaluate the cell's occupancy value. $p(m_t|x_t, z_t)$ is termed as inverse sensor model because the model reasons from effects to causes, where the model provides world information (m_t) from sensor measurements (z_t) instead of the other way around, $p(z_t|m_t, x_t)$. The cell's occupancy value is then converted into log odd notation in the occupancy grid map algorithm which is part of the map update step. The overall map update step implemented is described in our previous work [19]. To interpret cell's occupancy, selected sensor measurements with encoded cell's position were used as the neural network inputs. The following describes the input configuration of the neural network, $\mathcal{N}$:

- Four sensors measurements, z_t^k, $k \in \{1, 2, 3, 4\}$ that are closest to the cell, m_j.
- Encoded m_j's position using the distance, d_{m_j} and angle, θ_{m_j}.

Fig. 3 illustrates the measurements of d_{m_j} and θ_{m_j}. In the figure, array of infrared sensor on a mobile robot named Khepera III was used. This array configuration has nine infrared sensors, labelled as one until nine. The distance, d_{m_j} and angle, θ_{m_j} are determined by the sensor that is closest to cell m_j. In this illustration, sensor three has the closest distance. Thus, the position of cell m_j is encoded using the distance between m_j and sensor three and the angle of m_j to the same sensor. The four closest sensor measurements in this case will be sensor one, two, three and four, which are selected to be the input of neural network, $\mathcal{N}$.

At each time step, the sensor measurements and the predicted robot's state from motion model are preprocessed to determine the four closest sensor measurements, d_{m_j}, and θ_{m_j}. Hence, follows the configuration of the neural network, $\mathcal{N}$ input. The output of $\mathcal{N}$ is interpreted as probability of cell's occupancy $p(m_j|x_t, z_t)$. This process is repeated for all cells surrounding the Khepera III mobile robot, in order to build a local occupancy grid map, $\mathcal{G}$, with $n \times n$ cells as depicted in Fig. 3.

Fig. 4 shows part of the configuration of the neural network, $\mathcal{N}$. The input layer consists of input nodes that received the sensor measurements, encoded cell's position, and one bias node. There is no computation done on the input nodes but rather just to pass the inputs to the hidden layer. In the hidden layer, there are three neurons and one bias neuron. Bias neuron is used to provide every node in the adjacent layer with a constant value apart from the normal inputs that the node receives. Following normal practice, the bias for neural network, $\mathcal{N}$ is set to value 1. As for the output layer, since there is only one output which is the cell's occupancy value, thus there is only one neuron is in the output layer.

In neural network, each neuron in connected to all neurons in the adjacent layer as showed in Fig. 4. For each connection that connect from a neuron in one layer to the adjacent layer, there is a weight associated with the edge. Thus, in neural network, $\mathcal{N}$, there are $3{\times}7 = 21$ weights for the edges that connect the input layer to the hidden layer and $1{\times}4 = 4$ weights that connect the hidden layer to the output layer. Within these weights, four of them are associated with the bias neuron. Initially all the edges weights are random values.

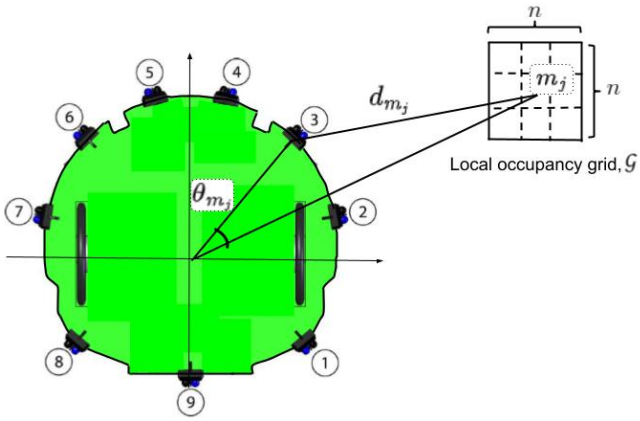

Fig. 3. The Position of Cell m_j is Encoded using d_{m_j} and θ_{m_j} Relative to the Robot.

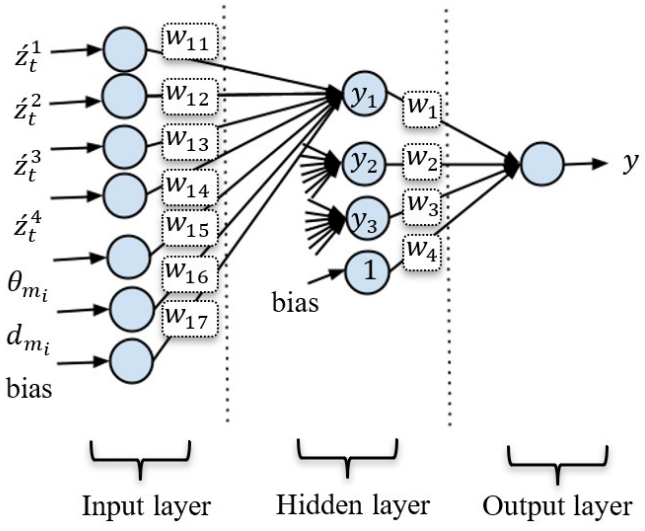

Fig. 4. Part of the Neural Network $\mathcal{N}$ Configuration.

Then, during the training process, for every set of input in the training dataset, each neuron in the hidden layer and the output layer in the neural network, $\mathcal{N}$ (excluding the bias neuron) is activated via an activation function. For example, for the first neuron in the hidden layer, the output of the neuron is obtained via equation (7).

$$y_1 = F(w_{11}z_t^1 + w_{12}z_t^2 + w_{13}z_t^3 + w_{14}z_t^4 + w_{15}\theta_{m_j} + w_{16}d_{m_j} + w_{17}) \quad (7)$$

Here, w_{xx} is referred to the weight associated with the input neuron as illustrated in Fig. 4. While F is an activation function that compute the output of that neuron, y_1. The same equation with respective weights is applied on all neurons in the hidden layer which will produce the rest of the outputs, y_2 and y_3. Then y_1, y_2, and y_3, becomes the input to the activation function for the neuron in the output layer using equation (8). Here, w_x, referred the weights on edges that connects the hidden layer to the output layer as seen in Fig. 4 as well.

$$y = F(w_1y_1 + w_2y_2 + w_3y_3 + w_4) \quad (8)$$

The output, y, is then compared with the output in the training data for that inputs. Then, the error is propagated back to the previous layer. From the error, the weights for the edges are adjusted. This process is known as backpropagation technique. To analyze the performance of the RBPF SLAM algorithm integrated with neural network $\mathcal{N}$, implementation of inverse sensor model with $\mathcal{N}$ will be tested and compared with the RBPF SLAM algorithm without neural network in simulation experiments.

G. Overall Grid-based SLAM Algorithm with Neural Network

Fig. 5 shows the overall algorithm for the RBPF SLAM algorithm with neural network, $\mathcal{N}$ implemented. In summary, the importance weight of the particles is the observation likelihood $p(z_t|x_t)$. The process and entities in red color describes the integration of neural network, $\mathcal{N}$. Initially, measurements from infrared sensors and sampled particles are preprocessed to build the inputs for $\mathcal{N}$. In the preprocess step, encoded position of cell, m_j, and the measurement of four closest sensors, z_t^k, $k \in \{1, 2, 3, 4\}$, are calculated. Then, these values are fed into $\mathcal{N}$ to build the local grid map, $\mathcal{G}$. Note that global occupancy grid map is maintained by each particle. Local map and global map are then used to calculate the importance weight of the particles by using the map matching algorithm.

The green boxes indicate selective resampling method described in section II-C. The dotted arrow lines represent the map update step which is executed after the particle's importance weight has been calculated. The final output is the density of robot's state estimate, x_{est} and its map, M, showed at the bottom part of the flowchart. The particles' density, x_{est} then updates the overall robot's trajectory, $x_{1:t}$.

III. Result and Analysis

The experiment was divided into two main parts. In the first part of the experiment, neural network, $\mathcal{N}$ was trained with the input configuration described in section II-F. Then, in the second part of the experiment, the resulting map and robot's state estimate are analyzed by comparing the results to the ground truth occupancy grid map and the ground truth robot's trajectory.

A. Training Neural Network, $\mathcal{N}$

A neural network tool in MATLAB named patternet was used to train the neural network, $\mathcal{N}$. At the end of the training, the MSE value obtained was 0.0965. Table I shows the weights of edges that connecting the input layer to the hidden layer where y_1, y_2, and y_3 are the neuron in the hidden layer. Table II shows the weights for the edges connecting the hidden layer to the output layer.

Next, the resulting map and robot's state estimate from the RBPF algorithm integrated with the neural network, $\mathcal{N}$ is analyzed by comparing the result to the ground truth data. The ground truth was obtained from the robot simulator platform used in the simulation experiment named Webots robot simulator [20].

B. Map Score and RMSE

Table III shows the performance of the RBPF with neural network, $\mathcal{N}$ SLAM algorithm in terms of map's accuracy and

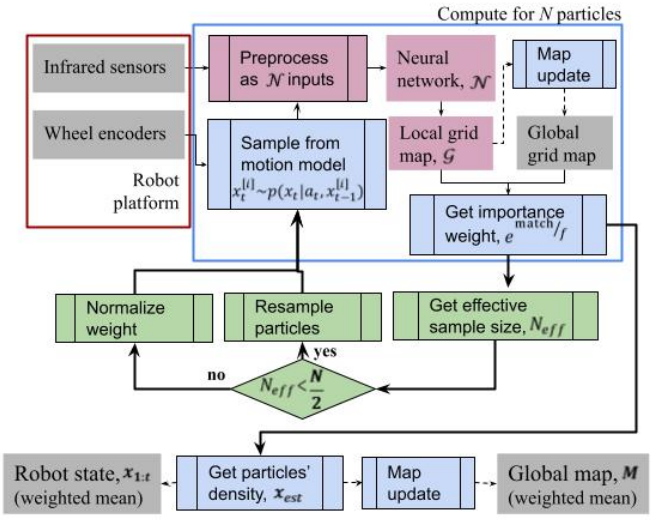

Fig. 5. Overall Algorithm for RBPF SLAM Algorithm with Neural Network, $\mathcal{N}$.

TABLE I. Weights for Edges Connecting the Input Layer to the Hidden Layer.

	z_1^t	z_2^t	z_3^t	z_4^t	θ_{m_j}	d_{m_j}	bias
y_1	-6.427	0.325	-0.207	0.166	0.529	0.003	1.608
y_2	-0.847	-0.615	-0.611	-0.23	1.992	-0.08	2.170
y_3	0.303	-5.525	-0.206	0.519	0.802	0.054	1.076

TABLE II. Weights for Edges Connecting the Hidden Layer to the Output Layer.

	y_1	y_2	y_3	bias
y	0.859	1.558	0.546	-1.325

TABLE III. MAP ACCURACY AND ROBOT'S STATE ESTIMATE ERROR OF RBPF SLAM ALGORITHM.

RBPF-SLAM	f_{occ}	f_{map}	RMSE(cm)
Without NN	0.2311	0.7012	34.84
With NN	0.7365	0.8491	9.42

robot's state accuracy. For the map state estimate, two methods of analysis were adapted; a fitness score for occupied cells, f_{occ}, and fitness score for all cells, f_{map}. While for robot's state accuracy, the root mean squared (RMSE) value was used as performance measure. This analysis methods is described in our previous work [19].

From Table III, it can be observed that the performance of robot's state estimate and map estimate are in agreement. Both analysis show that the accuracy of RBPF algorithm with neural network, $\mathcal{N}$ is better compared to RBPF algorithm without neural network in terms of robot's state estimate and map estimate accuracy. By adding neural network integration, the score of overall map estimate increases from 70.1% to 84.9%. While the RMSE of the robot's state estimate decreases from 34.84 cm to 9.42 cm.

C. Closing the Loop

Fig. 6 shows the test environment that was used in the simulation experiment with the Khepera III mobile robot in the middle. The rays of infrared sensors and ultrasonic sensors are also shown in the simulator. Fig. 7 shows the resulting map obtained from the simulation experiment. In the resulting map, three trajectories are shown. The robot's ground truth trajectory is the green line. The trajectory of robot state estimate from RBPF with neural network, $\mathcal{N}$ is shown in blue line, while the red line is the trajectory obtained from robot's odometry.

The visual observations of the resulting maps shows that when the robot revisited a known place (i.e. closing the loop) at approximately time step 1550, robot state estimate was able to have better estimate compared to odometry's trajectory. Fig. 7 shows robot's state estimate of RBPF with $\mathcal{N}$ algorithm in red rectangle managed to close the loop despite of robot's odometry begin to diverge from the actual path. This is a significant improvement because by using high variance sensor such as infrared sensor with sparse measurements, a grid-based SLAM algorithm with ability to correct robot's pose in indoor environment with static condition is obtained.

D. Effect of Resampling

As mention in section II-D of the methodology section, the resampling step was executed when the efficiency of particles, N_{eff} has become lower than half of number of particles, N (i.e. $N_{eff} < N/2$). In this experiment $N = 20$ was used. Thus, whenever N_{eff} has become lower than 10, the particles will be resampled and particles' weight will be equal again with value 1. When resampling occurred, the particles that has lower weights are eliminated probabilistically, and the higher weight particles are duplicated. Hence, in some cases, good particles (i.e. particles with better robot's state estimate) are deleted as well. This will cause the spikes in pose error of robots' state estimate.

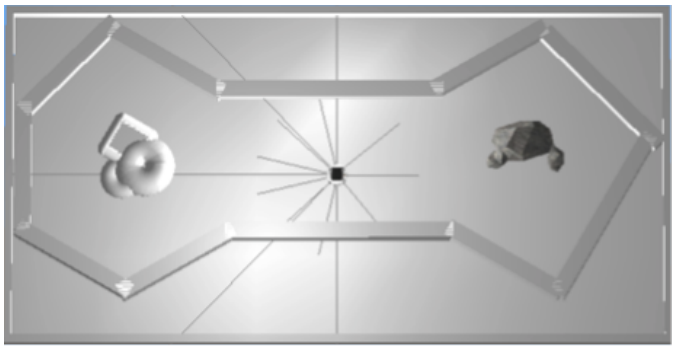

Fig. 6. Test Environment Created in Webots Simulator.

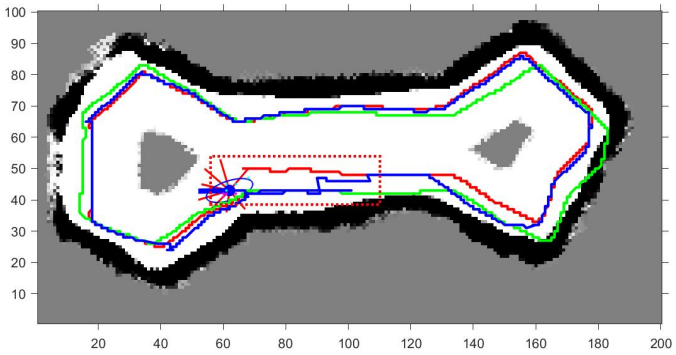

Fig. 7. Map and Trajectory of Robot's State of RBPF with $\mathcal{N}$ in Blue Line.

Fig. 8 shows the error of robot's state estimate of the RBPF algorithm with $\mathcal{N}$ along with the N_{eff} value at each time step throughout the robot's exploration. In this figure, some of the resampling steps are marked with black dashed line. From the figure, it can be observed that, whenever N_{eff} value drop below 10 and increased abruptly to 20, this is when the resampling steps occurred. If good particles were deleted, the pose error will increase. However, if bad particles were deleted, the robot's state estimate is better, hence the pose error decreases.

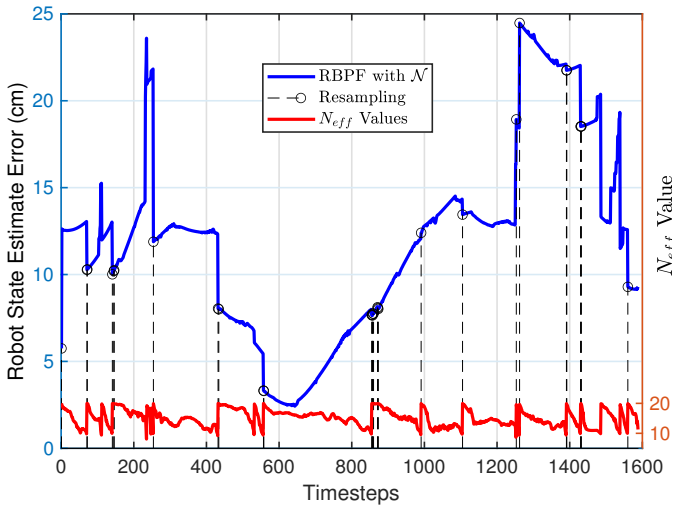

Fig. 8. The Pose Error of RBPF with $\mathcal{N}$ Align to the Values of N_{eff} at Each Time Step.

E. Number of Particles, N

The experiment is repeated multiple times and it is observed that by using 20 particles the consistency of a converge solution is not achieved. To achieve a more stable and consistent result, a higher number of particles were used. In the subsequent experiment, 100 particles were used with a longer travelled path, where the robot was set to navigate the same route twice in the test environment. Two methods of density extraction is compared which is weighted mean, as described in Section II-E and particle with the highest weight to represent the particles' density.

Fig. 9 shows the robot state estimate error for x_{est} using weighted mean, denotes as x_{mean} and the highest weighted particle, denoted as x_{max}. The robot finished navigate the environment in the first round at approximately time step = 200 and continue on the same path afterwards. The value of the time steps is lower in this experiment, as we increased the time interval between each sensor and odometry measurements taken. It can be observed that after time step 200, the state estimate of both x_{mean} and x_{max} gradually decrease. At this point, robot is basically computing localization using particle filter algorithm with known map. It is also noted that, after time step 200, x_{mean} and x_{max} has maintained below the error of robot state from odometry, x_{odom}. This is because, robot's state from odometry x_{odom} is increasing and as it diverges from the ground truth path.

Fig. 10 shows the resulting map obtained from particle with the highest weighted, x_{max}. The robot's trajectory uses the same color notation, which are green, blue and red lines referring to the ground truth trajectory, RBPF with $\mathcal{N}$ state estimate, and robot's odometry respectively. As the robot travelled further, the error in robot's state estimate accumulates. From the figure, it is observed that in the second loop of the robot's trajectory, the robot state estimate from RBPF with $\mathcal{N}$ (i.e. blue line) is able to localize itself within the map as it travelled through the revisited area. This has contained the error from accumulating further compared to odometry's trajectory (i.e. red line) that begin to diverge from the actual path. As mentioned before, this outcome is also reflected in

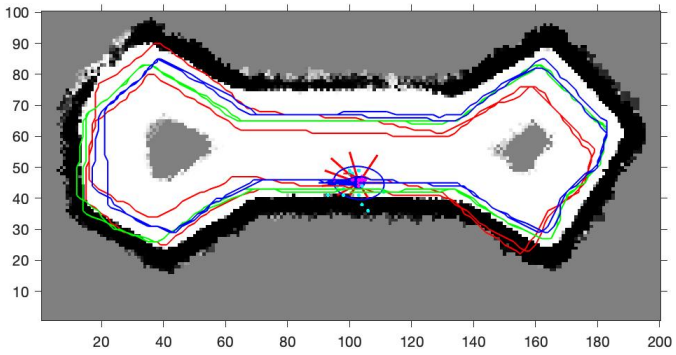

Fig. 10. Map of x_{max} of RBPF with $\mathcal{N}$ Algorithm with Robot's State Estimate and Actual Trajectory.

the pose error graph in Fig. 9, where after half way of the trajectory (i.e. approximately time step = 200) the robot state estimate error of both x_{mean} and x_{est} are below the error of odometry, x_{odom}.

The average RMSE of x_{mean}, x_{max}, and x_{odom} after completed the exploration is shown in the bar graph in Fig. 11. The bar graph shows the average RMSE for the distance error, denoted as x, y state, the x state, y state, and θ state. The x, y state uses the calculation of pose error of robot's From the bar graph, it can be observed that both density; x_{mean} and x_{max} have significantly lower average RMSE error from x_{odom} in red bar for all element of robot's state computed.

To test the consistency of the result, the experiment was repeated 25 times by using the same data set. The average RMSE of state estimate of all 25 trials is computed and shown in Fig. 12. From the bar graph it can be seen that the error of state estimate, x_{mean} and x_{max} are greater than odometry, x_{odom} for all states, except the θ state. This shows a rather different result compared to the one obtained in Fig. 11. It is concluded that, although RBPF with neural network $\mathcal{N}$ can achieve a lower error compared to odometry, x_{odom}, but it is not an entirely consistent result even with 100 particles. By using noisy sensor such as array of infrared sensors, a good

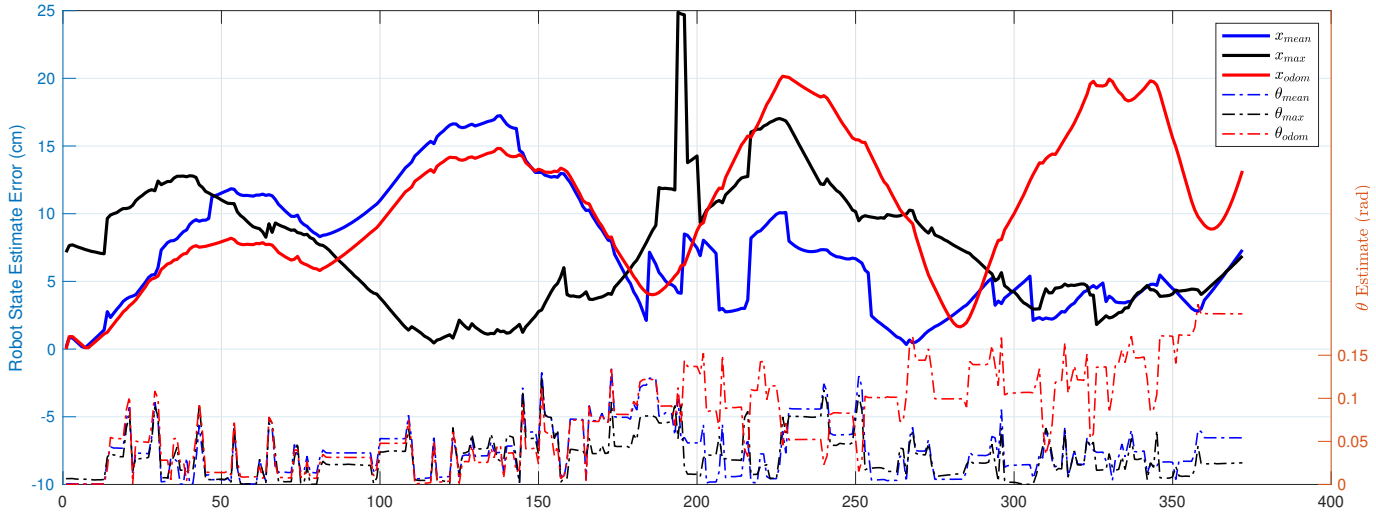

Fig. 9. The Pose Error of x_{mean}, x_{max}, and x_{odom} Align to the θ_{est} of Each Density.

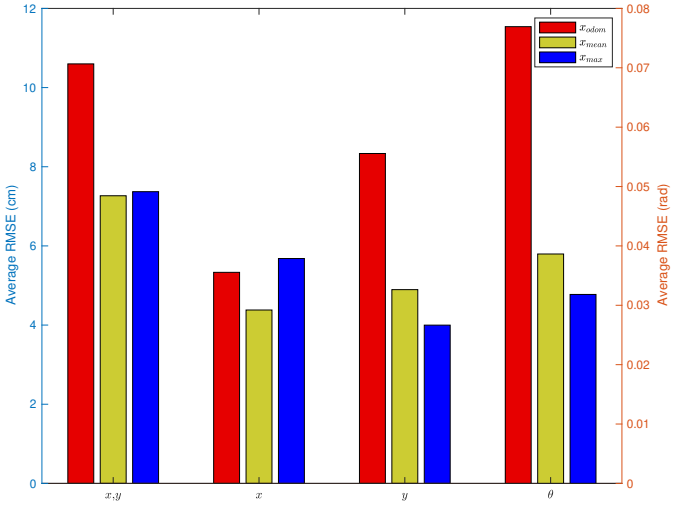

Fig. 11. Comparison of Average RMSE of x_{odom}, and Particles' Density x_{mean} and x_{max}.

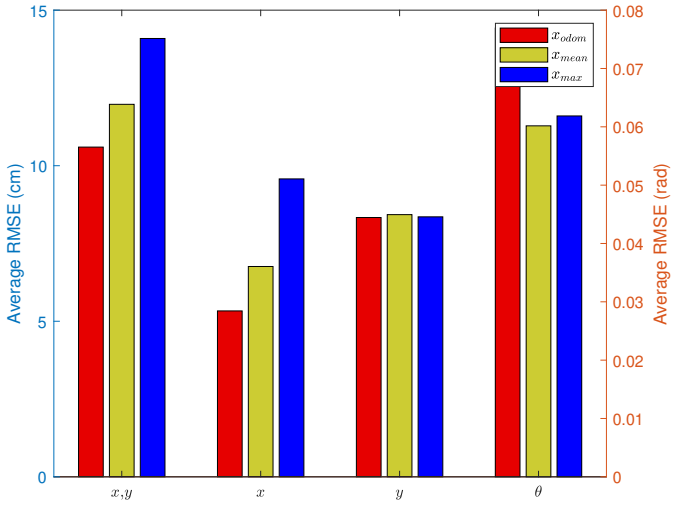

Fig. 12. Average RMSE of Robot's State Estimate after the Robot has Completed Navigation for 25 Trials of Experiment Conducted.

result can be obtained provided multiple trials were conducted.

IV. Conclusion

The first objective of this research was to develop grid-based SLAM algorithm with Rao-Blackwellized Particle Filter (RBPF) integrated with neural network for indoor environment. The algorithm was developed with motion model proposal distribution, a map matching function for particles' weight assignment and selective resampling method. The results show that integration of neural network has managed to increase the accuracy of map estimate and decrease the error of robot state estimate.

By using selected sensor measurements and encoded cell's position as neural network input, the neural network configuration is more robust in multiple ways. The neural network can be applied on robot platform with different sensors configuration. This is because the configuration does not require for sensors to be distributed equally apart or any specified angle for each sensor. Furthermore, by using cells encoded value (or polar coordinate) rather than Cartesian coordinate, local occupancy grid can be resized according to the sensor's maximum range without having to retrain neural network. This is because cell's position is encoded independent of the size of local map.

The consistency of the RBPF algorithm with neural network $\mathcal{N}$ is tested with multiple trials of the experiments. It is shown that the average RMSE of robot's state estimate exceed the RMSE of state computed from odometry even after the number of particles N is increased from 20 particles to 100 particles. This concludes to obtain an adequate map and state estimate from noisy sensors such as array of infrared sensors using the RBPF with neural network algorithm multiple trials should be conducted.

Acknowledgment

The authors would like to thank the Centre for Research and Innovation Management (CRIM), Centre for Telecommunication Research & Innovation (CeTRI), Fakulti Kejuruteraan Elektronik dan Kejuruteraan Komputer (FKEKK), Universiti Teknikal Malaysia Melaka (UTeM) and the Ministry of Higher Education Malaysia for the grant RACER/1/2019/TK04/UTEM//5 which has made this research possible.

References

[1] Yang Cong, Changjun Gu, Tao Zhang, and Yajun Gao. Underwater robot sensing technology: A survey. *Fundamental Research*, 1(3):337–345, 2021.

[2] Abhishek Gupta and Xavier Fernando. Simultaneous localization and mapping (slam) and data fusion in unmanned aerial vehicles: Recent advances and challenges. *Drones*, 6(4):85, 2022.

[3] Ravinder Singh and Kuldeep Singh Nagla. Comparative analysis of range sensors for the robust autonomous navigation–a review. *Sensor Review*, 40(1):17–41, 2019.

[4] Mubariz Zaffar, Shoaib Ehsan, Rustam Stolkin, and Klaus McDonald Maier. Sensors, slam and long-term autonomy: a review. In *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 285–290. IEEE, 2018.

[5] Stelian-Emilian Oltean. Mobile robot platform with arduino uno and raspberry pi for autonomous navigation. *Procedia Manufacturing*, 32:572–577, 2019.

[6] Francisco Quiroga, Gabriel Hermosilla, Gonzalo Farias, Ernesto Fabregas, and Guelis Montenegro. Position control of a mobile robot through deep reinforcement learning. *Applied Sciences*, 12(14):7194, 2022.

[7] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[8] Behzad Moshiri, Mohammad Reza Asharif, and Reza HoseinNezhad. Pseudo information measure: A new concept for extension of Bayesian fusion in robotic map building. *Information Fusion*, 3(1):51–68, 2002.

[9] Yun-Su Ha and Heon-Hui Kim. Environmental map building for a mobile robot using infrared range-finder sensors. *Advanced Robotics*, 18(4):437–450, 2004.

[10] Andres J Barreto-Cubero, Alfonso Gómez-Espinosa, Jesús Arturo Escobedo Cabello, Enrique Cuan-Urquizo, and Sergio R Cruz-Ramírez. Sensor data fusion for a mobile robot using neural networks. *Sensors*, 22(1):305, 2021.

[11] Maren Bennewitz, Cyrill Stachniss, Sven Behnke, and Wolfram Burgard. Utilizing reflection properties of surfaces to improve mobile robot localization. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4287–4292, 2009.

[12] Norhidayah Mohamad Yatim and Norlida Buniyamin. Development of Rao-Blackwellized Particle Filter (RBPF) SLAM algorithm using Low Proximity Infrared Sensors. In *The 9th International Conference on Robotics, Vision, Signal Processing & Power Applications*, 2016.

[13] Guillem Vallicrosa and Pere Ridao. H-slam: Rao-blackwellized particle filter slam using hilbert maps. *Sensors*, 18(5):1386, 2018.

[14] Cyrill Stachniss. *Exploration and mapping with mobile robots.* PhD thesis, 2006.

[15] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[16] Roland Siegwart and Illah R Nourbakhsh. *Introduction to Autonomous Mobile Robots*, volume 23. 2004.

[17] Christof Schröter, Hans-Joachim Böhme, and Horst-Michael Gross. Memory-Efficient Gridmaps in Rao-Blackwellized Particle Filters for SLAM using Sonar Range Sensors. In *EMCR*, 2007.

[18] G Grisetti. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *International Conference on Robotics and Automation*, number April, pages 32–37, Barcelona, Spain, 2005.

[19] N A Yatim, N Buniyamin, Z M Noh, and N A Othman. Occupancy grid map algorithm with neural network using array of infrared sensors. In *Journal of Physics: Conference Series*, volume 1502, page 12053. IOP Publishing, 2020.

[20] Olivier Michel. WebotsTM: Professional mobile robot simulation. *arXiv preprint cs/0412052*, 2004.