Volume 1, Issue VIII, August 2022, **No. 15, pp. 194-209**

Submitted 15/8/2022

Final peer review 22/8/2022

Online Publication 24/8/2022

THE KINEMATIC MODELING OF A FOUR MECANUM WHEEL ROBOT FOR ENVIRONMENT MAPPING AND NAVIGATION

[1]Princewill Chigozie Ene
Electrical and Electronic Engineering Department, Faculty of Engineering, Enugu State University of Science and Technology (ESUT), Enugu State, Nigeria.
princegozieene@gmail.com,eneh.princewill@esut.edu.ng

Abstract

This research presents the kinematic modeling of a four mecanum wheel robot for environment mapping and navigation. The study developed the kinematic model of the robot and then used Bayesian and kalman filter to improve the Simultaneous Localization and Mapping (SLAM) of 2D LIDAR sensor used for scanning. Q learning algorithm was used as an agent in reinforcement learning to control the action of the robot with respect to the SLAM data input. The performance of the learning algorithm when trained with 5000 episodes of the environment, achieved loss function approximately zero which implied that the episodes was correctly learnt. When deployed on the robot and then tested in an environment with strategically places obstacles, the result showed that the robot was able to learn as it navigated from each episodes until it was able to correctly avoid obstacles in the path and carry out its desired function.

Keywords: Kinematic, SLAM, LIDAR, Q learning, reinforcement learning, loss function

1. INTRODUCTION

The fundamental functions of the autonomous mobile robot are mapping, navigation, and localization. As the robot executes its desired tasks under command, it needs to do so at a rapid speed, collect dynamic environment information, including its current position, and make a decision to safely avoid static and dynamic obstacles to reach the goal within the prescribed time. Interaction with the environment such as grabbing, pushing, lifting and manipulating objects are the important capabilities of robotic systems as it avoids collisions and navigate in the workspace to reach goal location. The kinematic coordination, dynamic interaction and coupling of the different units must be considered in other to control robot’s functionalities, mobility, and manipulation. [1] Stated three common types of mobile robots; (i) vehicles fitted with wheels similar to general vehicles (automobile); (ii) two parallel wheels with one caster wheel and (iii) omnidirectional wheels.

In this work, an omnidirectional four-wheel robot was modeled and was later used for obtaining the graphical representation of an indoor environment. An omnidirectional mobile robot is a kind of holonomic robot that has the ability to move in translational and rotational simultaneously and independently [2]. According to [2], holonomic or mecanum wheel platform has the ability to simultaneously and independently achieve rotational and translational motion capabilities. An omnidirectional robot is a robot that can travel along the ground plane (x, y) in any direction without considering the actual orientation of the robot around its vertical axis. The ability of the wheels to move in more than just one direction determines the level of maneuverability of a mobile robot with Swedish mecanum or spherical wheels. Most robotic platforms will have an omnidirectional and high level of maneuverability attributes.

The forward and Inverse kinematic for omnidirectional mecanum wheels in relation to its platform was developed in [3]. The experimental analytical results showed that 8 different motions are possible without changing the robot's orientation. It was stated in [4] that most omnidirectional wheels are very sensitive to the road conditions which limits their operational performances. Furthermore, it was explained that each wheel attached to a mobile robot has omnidirectional characteristics on a plane surface [4].The angle rollers of the mecanum wheel transform a portion of the force in the rotational direction of the wheel into a force common to the direction of the wheel, as shown in figure 1. Depending on each individual wheel direction and speed, the resulting combination of all these forces creates a maximum force vector in any desired direction, allowing the platform to move freely in the direction of the resulting force vector, without altering the wheels themselves.

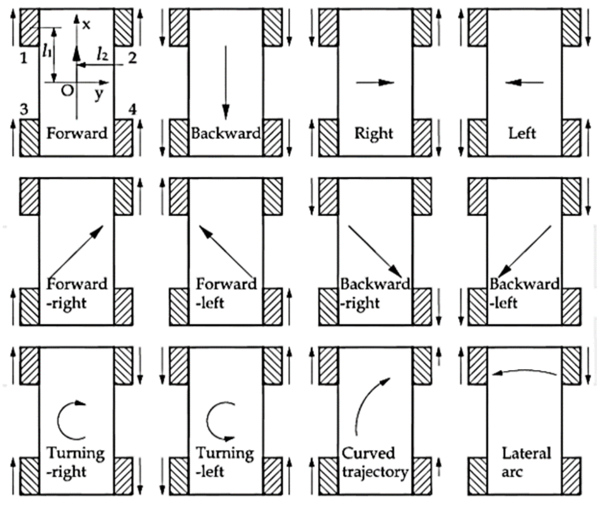


Figure 1: Robot motion according to the direction and angular speed of the wheels.

The desired directional movement is achieved by the combination of different wheels rotation and speeds. For all applications of mobile robots, the necessary things needed to be done are mathematical modeling, path planning, fluctuation elimination, the use of appropriate mechanisms and the use of appropriate control algorithms [5]. Analysis of robot stability and fluctuation elimination through the addition of passive rollers on the wheels and Moment Height

Stability (MHS) measurement was presented [6, 7]. [8] Also presented the design of a circular four-wheeled omnidirectional mobile Robot with the speed control implementation that was based on a PID control algorithm. In related research, a singularity free dynamic operation space dynamic modeling approach based on Lagrange's form of the D' Alembert principle was presented in [9]. It could be seen from the literature that Omnidirectional robot with mecanum wheels surfer limitations such as slippage and lateral movement. These limitations impede their ability to achieve proper control for navigation purposes

1.1 Problem statement

Limited reasoning and the representational ability for unstructured environmental models in robot navigation impede a completely reactive approach which causes a path to the goal position not always guaranteed. Hence, path planning is a challenging operation due to uncertainty like dynamic environment. Base on the factor stated above, deep reinforcement learning was proposed in this paper to make the learning agent learn the best policy to accumulate the most positive reward over time. This will allow the robot to learn how to find and follow the best path from the starting location to the target location without prior environmental knowledge.

2. METHODOLOGY

There are basically two mathematical models of the robot that was formulated; the kinematic model and dynamic model of holonomic robot structure but here we will be dealing only on the kinematic model. The kinematic model of a robot describes the motion of the robot mechanical structure (fixed and movable units in the robot) without considering the mass of each of the units or the forces that caused the motion of the units. While the dynamic model describes the state of the robotic system variables as the operation evolves with time. Also, robot dynamics explains the relationship between the forces acting on a robotic structure with respect to the acceleration produced by the structure. However, it has been a challenge in understanding the kinematics and the dynamics of a holonomic four-wheel mobile robot platform due to the complex wheel-ground interaction and the kinematic constraints imposed on the platform by the four mecanum wheels. The robot platform in this work is an omnidirectional mobile platform that provides 3 degrees of freedom (DoF) in 2D space. The robot used 2D LIDAR scanner to Simultaneously Localize and Map (SLAM) the environment, then reinforcement learning was used to control the behavior based on deep Q-learning algorithm. This was implemented using high programming language and the performance evaluated.

2.1 Four-Wheel Mobile Robot Kinematics

The wheel mobile robot kinematics is of three types [10]; Internal kinematics, External kinematic, and forward and inverse kinematics. The internal kinematic defines the relationships between external variables of the systems, such as the rotation of the wheel and robot movement. External kinematics defines the robot's position and orientation with respect to some coordinate frame of reference. Forward and reverse kinematics or direct kinematics solves the robot's state depending on its inputs. The inputs can be wheel speeds, movement of joints, steering of wheels, etc. A four-wheel mobile robot's steering is based on controlling the robot's four left and right differential wheel relative velocities [11].

The input to drive a mobile robot is the speed of the right and left wheels and the robot's orientation. The pose of a robot plane in an initial frame or global coordinate frame (X_G, Y_G) can be represented in figure 2.

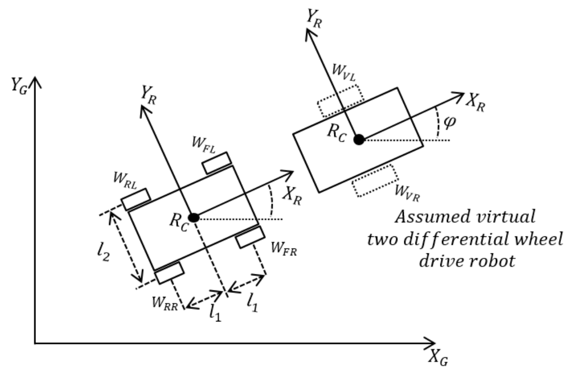


Figure 2: Kinematics model of Four-wheel Mobile Robot in Global Coordinate Frame

The forward kinematic model of the four-wheel robot was done to determine the robots' location and orientation through the wheels' rotation measurement obtained from the optical encoder. The kinematic modeling was simplified by treating the differential four-wheel drive of the robot as a two virtual wheel as shown in figure 2. A differential drive means the robot can change direction by varying the relative rate of rotation of each wheel without the need for additional steering motion.

The forward kinematic model parameters of a four-wheel robot are;
X_C - robot geometric centre X position
Y_C - robot geometric centre Y position
X_r - robot local X axis that determines the front of the robot
φ - robot angular position
R_C - robot geometric center
W_{FR} - front right wheel
W_{RL} - rear left wheel
W_{RR} - rear right wheel
W_{FL} - front left wheel
W_{VR} - virtual right wheel
W_{VL} - virtual left wheel
l_1 - distance between the robot center and front/rear wheels
l_2 - distance between robot left and right wheels

The following assumptions were made while developing the forward kinematic model;

1. The robot's mass centre is at the robot frame's geometric centre.
2. The two wheels rotate at the same speed on each side of the robot so that the four-wheel robot can be viewed as a differential two-wheel drive robot.
3. The wheels of the robot make firm contact with the horizontal ground surface and can only move in x and y plane.
4. There is no tire deformation.

The control vector of the robot that describes the robot's center of gravity (CoG) in terms of the global coordinate frame can be expressed as;

$$q(t) = \begin{bmatrix} x(t) \\ y(t) \\ \varphi(t) \end{bmatrix} \tag{1}$$

Where
$x(t)$ – is the x- axis position of the robot within the global coordinate frame
$y(t)$ – is the y- axis position of the robot within the global coordinate frame
$\varphi(t)$ – is the orientation or heading of the robot w.r.t the x- axis global coordinate frame

The left and right wheels have radius$r$, and they are separated by a distance l_1 from each other. Their angular positions are described by θ_R and θ_L.The φ is the robot orientation angle measure from the $X_G - axis$ of the global frame to $X_R - axis$ of the local frame of the robot. Therefore, the vector representation of the robot angular position w.r.t the global frame is;

$$q(t) = [x\, y\, \theta_R \theta_L]^T \tag{2}$$

The vector of generalized velocities is

$$\dot{q} = \left(\dot{X}, \dot{Y}, \dot{\theta}\right)^T \tag{3}$$

The linear velocity of the robot moving in a plane can be expressed in terms of a moving robot or moving frame as

$$v \triangleq \left[v_x, v_y, \omega_i\right]^T \epsilon\, \mathbb{R}^3 \tag{4}$$

Where v_x and v_y are the longitudinal and lateral velocity of the mobile robot [12].
Following the initial frame coordinates (X_G, Y_G), the orientation θ, combined with the robot velocities in both the local frame and initial frame as given in equation 3.1 and equation 3.4, the kinematic equation of motion using the rotation matrix approach is written as;

$$\dot{q}\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & sin\,\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_i \end{bmatrix} \tag{5}$$

where $\dot{q} \in \mathbb{R}^3$ is the generalized velocity vector and ω_i is the four wheels angular velocity of the robot, thus, $i = 1, 2, 3, 4$.

Let the linear speeds of the right virtual wheel (v_R) and the left virtual wheel (v_L) be the control parameters for the robot motion operation. The new position (X_R, Y_R, φ) of the robot after some time Δt is obtained by solving for the angular speed ω of the mobile robot center of gravity (*CoG*) and the angular position φ of each virtual wheel as the average of its corresponding wheels as follow;

$$\omega W_L = \frac{\varphi W_{FL} + \varphi W_{RL}}{2} \tag{6}$$

$$\omega W_R = \frac{\varphi W_{FR} + \varphi W_{RR}}{2} \tag{7}$$

$$\varphi W_{VL} = \frac{\varphi W_{FL} + \varphi W_{RL}}{2} \tag{8}$$

$$\varphi W_{VR} = \frac{\varphi W_{FR} + \varphi W_{RR}}{2} \tag{9}$$

For the assumed virtual wheels, the linear speeds for each of the wheels are;

$$v_R = \omega W_R \times r \tag{10}$$

$$v_L = \omega W_L \times r \tag{11}$$

Where r is the radius of the wheel.

The robot's angular position and speed are written as;

$$\varphi = (\omega W_R - \omega W_L)\frac{r}{l_2} \tag{12}$$

The robot speed for x and y components are;

$$\dot{x}_c = \left(v_L + \dot{\varphi}\frac{l_2}{2}\right)\cos\varphi \tag{13}$$

$$\dot{y}_c = \left(v_L + \dot{\varphi}\frac{l_2}{2}\right)\sin\varphi \tag{14}$$

The position of the mobile robot is obtained by integrating the x and y components of the robot's speed as;

$$x_c = \int_0^t \dot{x}_c\, dt \tag{15}$$

$$y_c = \int_0^t \dot{y}_c\, dt \tag{16}$$

2.2 Simultaneous Localization and Mapping (SLAM)

SLAM is a computational approach of constructing and updating the map of an unknown environment while keeping track of the robot position (pose) inside the map as it moves around and explores the environment through a collection of information using the onboard sensors. The map used for localization in this study was generated by the SLAM method. The idea was to enable the robot identify everything along its propagation path (Environment) map out its environment and at the same time. This was achieved used a 2D LIDAR and odometer sensors respectively. The LIDAR scan for the environment while the odometer was used to map the detected images. Bayesian filter was used to process the data captured while kalman filter [18] was used to update the output. The workflow of the SLAM process was presented in the figure 3, while the Bayesian algorithm was presented in algorithm 1;

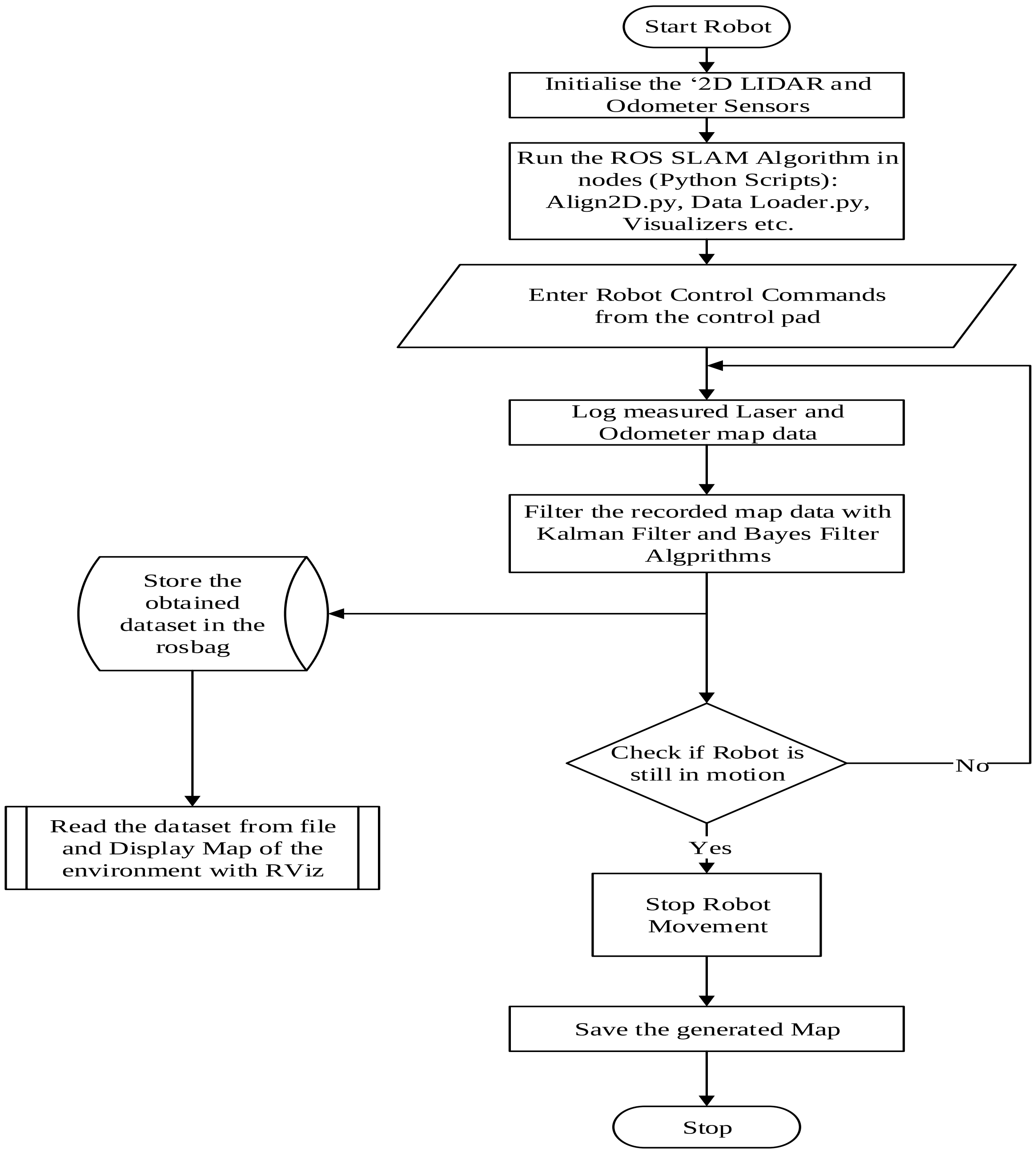

Start Robot
Initialise the '2D LIDAR and Odometer Sensors
Run the ROS SLAM Algorithm in nodes (Python Scripts): Align2D.py, Data Loader.py, Visualizers etc.
Enter Robot Control Commands from the control pad
Log measured Laser and Odometer map data
Filter the recorded map data with Kalman Filter and Bayes Filter Algprithms
Store the obtained dataset in the rosbag
Read the dataset from file and Display Map of the environment with RViz
Check if Robot is still in motion
No
Yes
Stop Robot Movement
Save the generated Map
Stop

Figure 3: Flow Chart for SLAM Operation

Algorithm 1: Recursive Bayes Filter

```
1:  function BAYES_FILTER ($bel(x_{t-1}), d, z_t, u_t,$);
2:  $\eta = 0$
3:  if $d$ is a perceptual data item $z_t$ then
4:          for all $x_t$ do
5:                  $\overline{bel}(x_t|u) = \int p(x_t|\, u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$;
6:                  $\eta = \eta + \overline{bel}(x_t|u)$
7:          for all $x_t$ do
8:                  $\overline{bel}(x_t|u) = \eta^{-1}\overline{bel}(x_t|u)$
9:   else if $d$ is an action data item $u$ then
10:         for all $x_t$ do
11:  $bel(x_t|u) = \sum p(z_t|x_t)\, bel(x_t)$;
12:  end for
13:  return $bel(x_t)$
14:  end function
```

3. REINFORCEMENT LEARNING ALGORITHM

Having developed the robotic system model in figure 2 and the SLAM model in figure 3, the control system which used the information mapped to take decision was achieved with Reinforcement Learning (RL). RL is a semi-supervised learning model in machine learning that enables an agent to learn by interacting with its environment, using feedback from its own actions and experiences to maximize its total rewards. The major components of a RL agent are [20];

i) A Policy
ii) A Value Function
iii) A discount factorγ
iv) A Model

Policy (π) is the agent's best action by experience (behavior) based on how it is reinforced to learn by acting in the environment. The policy establishes the states $(s \epsilon S)$ of the environment into actions$(a \epsilon A)$, and it is improved in an RL problem on every step the agent moves in the environment state. At the end of the total actions performed for a given task, the cumulative sum of the immediate rewards (r) receive is returned (G) and used to optimize the policy. The effect of action (a) in a given state (s) is known as the reward $r(s, a)$ and the long-term reward from a state (a) after taken series of actions following a policy (π) is denoted by $G(s, a)$. A value function expresses how good it to be in a particular state is, and how good is it to take a particular action. Value function tells the agent much reward to expect if it chooses a particular action in a particular state. It's a prediction of expected future rewards used to evaluate the outcome of states, therefore enabling the agent to select between different actions. The discount factor (γ) informs the agent of how much it should care about rewards at the moment in order to reward properly in the future. As γ tends to zero (0), that means the agent cares about the first reward. While γ moves towards one (1), that means the agent cares about all future rewards. A model is the agent's representation of its environment, i.e. how the agent sees the environment. A model predicts what the environment will do next.

3.1 Q-Learning algorithm

Q-Learning algorithm is defined as an off-policy and a model-free RL algorithm that progressively processes the transition samples [19]. Q-learning seeks to discover the best action to perform in a given state thereby learning a policy that produces a maximum total reward. It is an off-policy because the Q-learning function does not depend on action within the current policy, thus Q-learning takes random action. In this section, model-free RL using a Q-learning algorithm is applied to mobile robot navigation where the robot is assumed to act in a stochastic environment. The mobile robot was made to sequentially select actions from a sequence of time steps so as to maximize its collective reward. The robot navigation was model as MDP where a state-space S, an action space $\mathcal{A}$, and a transition dynamic distribution $P(s_{t+1}|s_t, a_t)$ that satisfies the Markov property $P(s_{t+1}|s_1, a_1, \dots, s_t, a_t) = P(s_{t+1}|s_t, a_t)$ for all trajectory $s_1, a_1, s_2, a_2 \dots, s_t, a_t$within the state-action space, and a reward function $r: \mathcal{S} \times \mathcal{A} \longmapsto \mathbb{R}$ were considered. Trajectory of states, actions and rewards $s_1, a_1, s_2, a_2 \dots, s_t, a_t$, were produced by a stochastic policy $\pi(s_t, a_t) = P(a_t|s_t)$ over $\mathcal{S} \times \mathcal{A} \times \mathbb{R}$.

The Q-value is given as,

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \overset{max}{\underset{a}{}} Q(s_t', a_t) \tag{17}$$

Where

$Q(s_t, a_t)$ is the learned action value (Q-value).

γ is the discount factor that is used to balance immediate reward and future reward.

The above equation states that the Q-value generated by existing in the state s_t and taking an action a_t is the sum of the immediate reward $r(s_t, a_t)$and the highest Q-value possible from the next state s'. Gamma γ, the discount factor which regulates the influence of immediate or future rewards. The discount factor can be set between 0 and 1. Setting it to a value of less than 0 applies the algorithm to take future rewards which make it converge faster.

Again, $Q(s_t', a_t)$ depends on $Q(s_t'', a_t)$ which has a coefficient of gamma squaredγ^2. So, the current Q-value comes from the Q-values of future states as shown in equation 18;

$$Q(s_t, a_t) \rightarrow \gamma Q(s_t', a_t) + \gamma^2 Q(s_t', a_t) \dots \dots \dots \ \gamma^{\mathrm{n}} Q(s_t'', a_t) \tag{18}$$

Altering the value of gamma will reduce or increase the influence of future rewards. Since this is a recursive equation, arbitrary assumptions for all q-values were made so that it will converge to the optimal policy. Practically, this is realized as a Q-learning update strategy defined as,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \overset{max}{\underset{a}{}} Q(s_{t+1}, a_t) - Q(s_t, a_t) \right] \qquad (19)$$

Where α is the learningstep rate taken to update the estimation of $Q(s_t, a_t)$. The learning rate or step size is set between 0 and 1 to simply specify the degree to which the newly acquired information supersedes old information. Setting it to 0 will make the Q not to update and learning cannot be achieved. While if the learning set is set to value towards 1, such as 0.9 means that learning will be done faster. The discount factor also can be set between 0 and 1, but setting it to a value of less than 0 makes the algorithm to converge faster. The table 1 presented the reward function for the Q learning and the Q-learning algorithm for the Q-value update rule is shown in the figure 4;

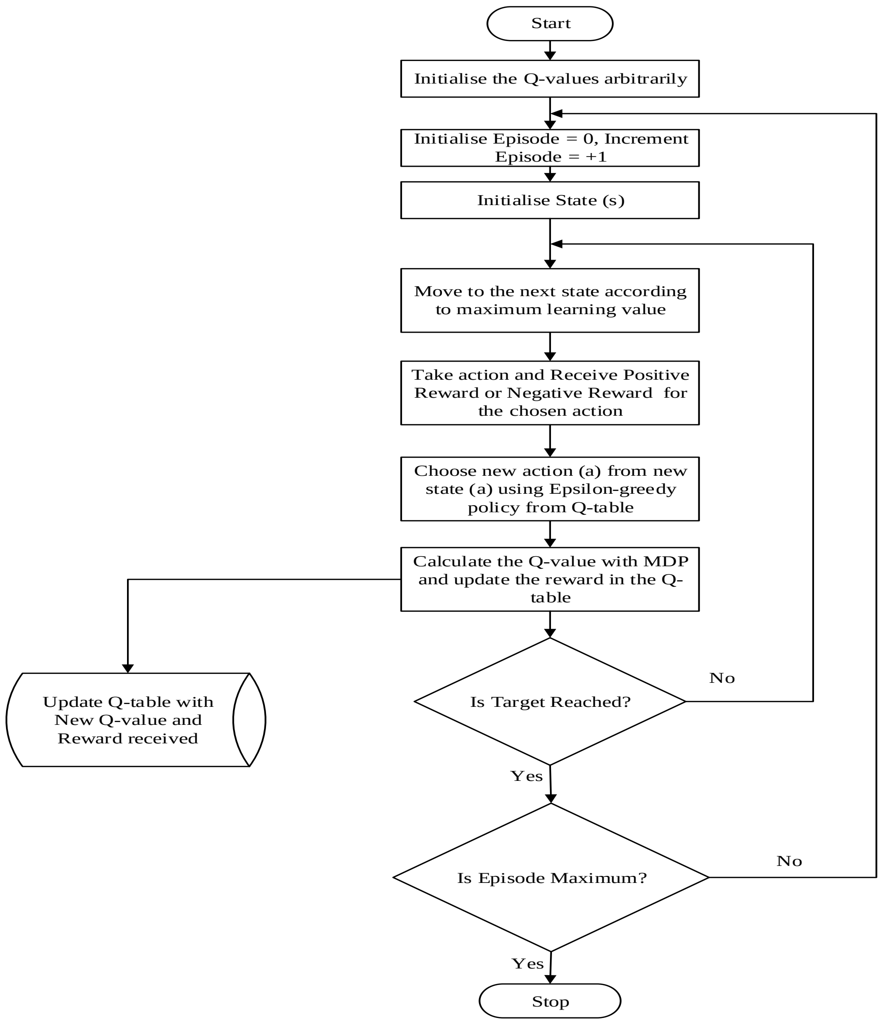

Start
Initialise the Q-values arbitrarily
Initialise Episode = 0, Increment Episode = +1
Initialise State (s)
Move to the next state according to maximum learning value
Take action and Receive Positive Reward or Negative Reward for the chosen action
Choose new action (a) from new state (a) using Epsilon-greedy policy from Q-table
Calculate the Q-value with MDP and update the reward in the Q-table
Update Q-table with New Q-value and Reward received
Is Target Reached?
No
Yes
Is Episode Maximum?
No
Yes
Stop

Figure 4: Model of the Q learning

Table 1: Reward Function Lookup table

State Transition	Criteria	Reward Function Value (r)
Other State → Target State		1
Safe State → Cosy State		-0.2
Cosy State → Safe State		0.8
Risky State → Cosy State		0.6
Cosy State → Risky State		-1
Risky State → Risky State (moving too closer to Obstacle)	$d_{obs} \leq d_{mim}^{t} \leq d_{mim}^{t-1} - 2$	-0.2
	$d_{mim}^{t} = d_{mim}^{t-1} - 1 \leq d_{obs}$	-0.2
	$d_{mim}^{t} = d_{mim}^{t-1} - 2 \leq d_{obs}$	-0.5
	$d_{mim}^{t} = d_{mim}^{t-1} - 3$	-1
Non-Safe State → Non-Safe State (Avoiding Obstacle)	$d_{obs} \leq d_{mim}^{t}$	0.7
	$d_{mim}^{t} \geq d_{obs}$	0.5

4. SIMULATION AND RESULTS

The algorithms developed were implemented on the robot using Simulink and the performance evaluated. The SLAM model was used to configure the 2D LIDAR sensor for the data acquisition while the Q learning was used to take decision and control the kinematics of the robot in an environment. The environment used has a dimension of 200×200 (m^2) and the obstacles were randomly placed in the environment. The robot was not given prior knowledge of its environment. The robot is indicated as a blue rectangular box located at $(180, 20)$ in the map, and the target is placed at$(180, 40)$. The aim is to navigate the robot starting from the initial robot position to the target position at a fixed speed of $3.12\ m/s$. This navigation was done by finding the optimal path from the initial point to the target position that is collision-free.

The self-learning process containing 500 moving steps per episode was conducted for 600 learning episodes. In every episode, there is a different configuration of random obstacle positions, and the robot tries to take as many steps as it can, learning every step from the received rewards. The training phase stops when all the learning episodes have finished. The parameter values used for the navigation simulation are as below;

- Learning Rate = *0.7*
- Discount Factor = *0.81*,
- Exploration Constant =*0.9*.

The robot was trained in three different environments to expose it to new challenges that would cause the robot to avoid being trapped into an endless loop after learning the environment. The result of the learning process was presented in the figure 5;

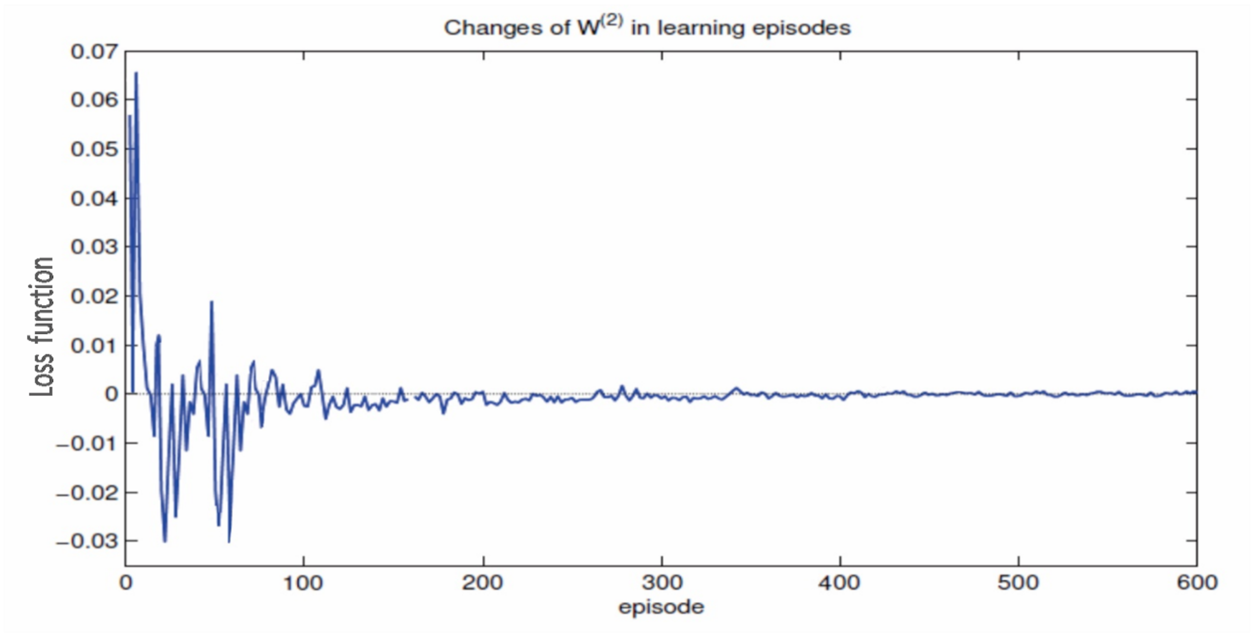


Figure 5: The performance of the Q learning process

From the result in figure 5, it was observed that the Q learning was able to learn the environment correctly as the loss function recorded was approximately zero which implied good training process. The performance result of the SLAM was presented in figure 6;

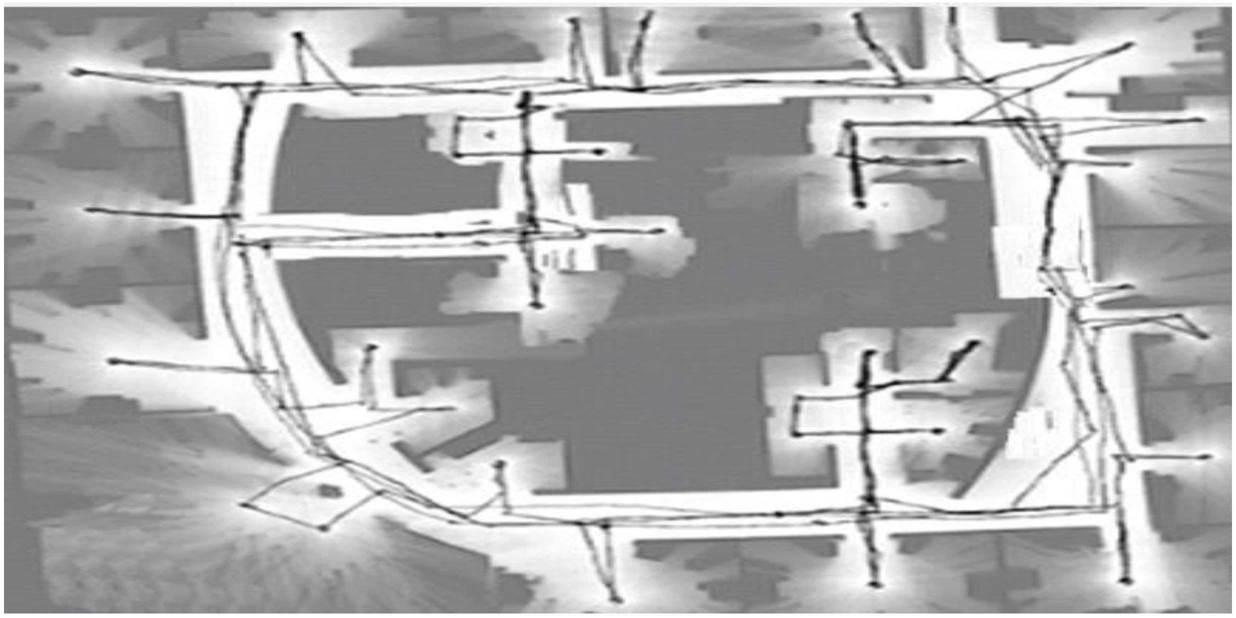
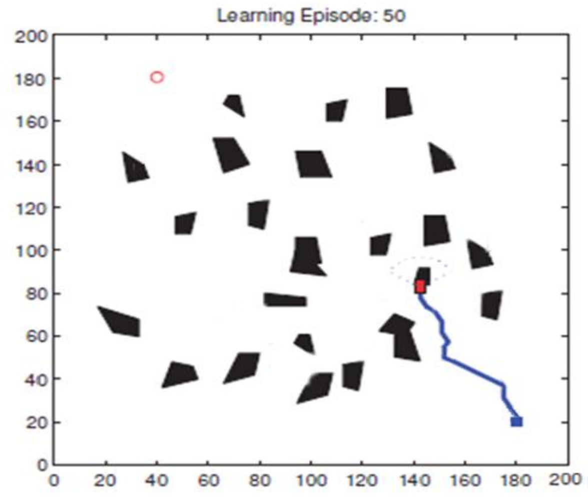
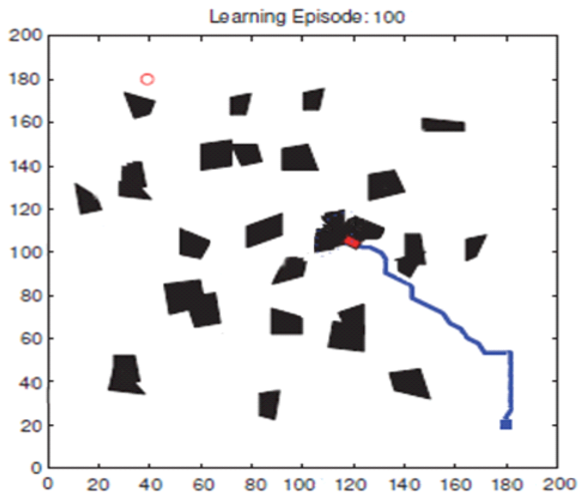


Figure 6: Indoor Map Visualization using SLAM

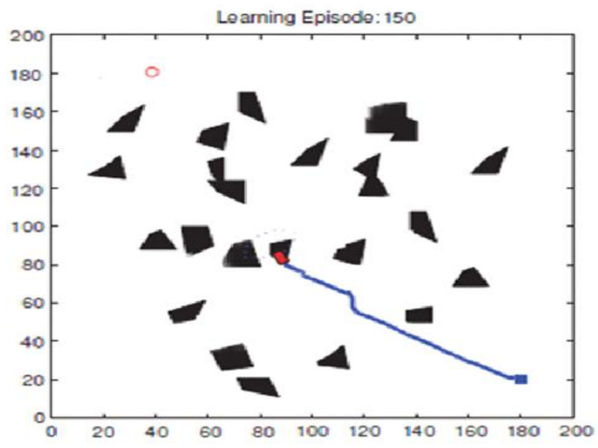
The figure 6 presented the result of the SLAM used for the mapping of the environment and localization to detect free path for navigation. The result showed that the Bayesian filter algorithm was able to allow the scanner to map and read clearly in 2D the environment. The data collected from the SLAM process was used by the reinforcement learning control system for specific action to guide the navigation of the robot from obstacles as shown in the figure 7.



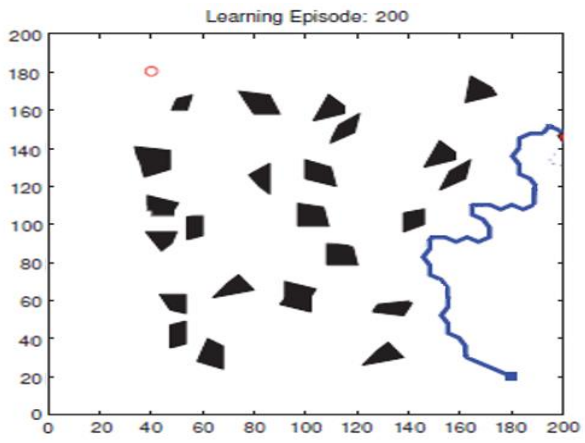
(a)



(b)



(c)



(d)

Figure 7: Performances of the robot with the learning algorithm at various episodes

The figure 7 presented the performance of the robot as it navigated and learn from various episodes as in (a, b, c and d) respectively. The first step performed by the robot during navigation was to identify its current state and whether the current state is a safe state or not. If the state is not safe, the robot adopts a control policy; else, it changes its direction towards the goal and makes a step toward the target. Before using a control policy from the Q-table 1, the robot generates all state-action Q-value pairs ever possible so that it can greedily select the action with the highest Q-value. It then find its next state from the selected action and continues this process of action to select and move until it either reaches its target point or collides with an obstacle or hits the boundary wall.

5. CONCLUSION

This paper presents the mobile robot with improved SLAM performance using Bayesian and kalman filter. The robot was developed with reinforcement learning trained with Q learning algorithm as the agent. The performance when tested showed that the robot was able to learn as it navigates based on the Q learning model and give rewards based on the table 1 and update till it completely understood the environment and performed its task successfully.

6. References

[1] Djebrani, Salima, Benali, Abderraouf and Abdessemed, Foudil, "Modelling and control of an omnidirectional mobile manipulator" *International Journal of Applied Mathematics and Computer Science*,2012, vol.22, no.3, pp.601-616. https://doi.org/10.2478/v10006-012-0046-1

[2] Pin FG, Killough SM.,"A new family of omnidirectional and holonomic wheeled platforms for mobile robots". *IEEE transactions on robotics and automation*. 1994 Aug;10(4):480-9.https://www.osti.gov/servlets/purl/10157906.

[3] Taheri, H., Qiao, B., & Ghaeminezhad, N., (2015). "Kinematic Model of a Four Mecanum Wheeled Mobile Robot". *International Journal of Computer Applications*,2015,*113*(3), 975–8887. https://doi.org/10.5120/19804-1586

[4] Chung, J. H., Yi, B.-J., Kim, W. K., & Han, S.-Y.,"Singularity-Free Dynamic Modeling Including Wheel Dynamics for an Omni-Directional Mobile Robot with Three Caster Wheels". *The International Journal of Control, Automation, and Systems*, 2008, *6*(1), 86–100. https://doi.org/10.1109/ROBOT.2003.1241647

[5] Mardany, A., & Ebrahimi, S.,"Dynamic modeling and construction of a two-wheeled mobile manipulator, part II: Modified obstacle climbing*". In International Conference on Robotics and Mechatronics, ICROM*, 2015, pp. 170–175. https://doi.org/10.1109/ICRoM.2015.7367779

[6] Watanabe, K., Shiraishi, Y., Tzafestas, S. G., Tang, J., & Fukuda, T.,"Feedback Control of an Omnidirectional Autonomous Platform for Mobile Service Robots". *Journal of Intelligent and Robotic Systems: Theory and Applications*,1998,*22*, 315–330. https://doi.org/10.1023/A:1008048307352

[7] Williams, R. L., Carter, B. E., Gallina, P., & Rosati, G., "Dynamic model with slip for wheeled omnidirectional robots". *IEEE Transactions on Robotics and Automation*, 2002,*18*(3), 285–293. https://doi.org/10.1109/TRA.2002.1019459

[8] Huang, L., Lim, Y., Li, D., & Teoh, C. E. L., "Design and analysis of a four-wheel omnidirectional mobile robot". In *Proc. of the 2nd Int. Conf. on Autonomous Robots and Agents*, 2004, pp. 425–428.

[9] Chung, J. H., Yi, B.-J., Kim, W. K., & Han, S.-Y., "Singularity-Free Dynamic Modeling

Including Wheel Dynamics for an Omni-Directional Mobile Robot with Three Caster Wheels". *The International Journal of Control, Automation, and Systems*,2008, *6*(1), 86–100. https://doi.org/10.1109/ROBOT.2003.1241647

[10] Klančar, G., Zdešar, A., Blažič, S., & Škrjanc, I. (2017). *Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems*. Butterworth-Heinemann. ISBN 9780128042045. https://doi.org/10.1016/B978-0-12-804204-5.00001-9.

[11] Wang, T., Wu, Y., Liang, J., Han, C., Chen, J., & Zhao, Q. (2015). Analysis and experimental kinematics of a skid-steering wheeled robot based on a laser scanner sensor. *Sensors (Switzerland)*, *15*(5), 9681–9702. https://doi.org/10.3390/s150509681

[12] Kozlowski, K., & Pazderski, D. (2004). Modeling and control of a 4-wheel skid-steering mobile robot. *International Journal of Applied Mathematics and Computer Science*, *14*(4), 477–496. https://doi.org/10.1016/j.compag.2011.12.009

[13] Tzafestas, S. G. (2014). *Introduction to Mobile Robot Control*. *Introduction to Mobile Robot Control*. https://doi.org/10.1016/B978-0-12-417049-0.00001-8

[14] Caracciolo, L., Luca, a. De, & Iannitti, S. (1999). Trajectory tracking control of a four-wheel differentially driven\nmobile robot. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. https://doi.org/10.1109/ROBOT.1999.773994

[15] Sarkar, N., & Kumar, V. (1994). Control of Mechanical Systems With Rolling Constraints: Application to Dynamic Control of Mobile Robots. *The International Journal of Robotics Research*, *13*(1), 55–69. https://doi.org/10.1177/027836499401300104

[16] Caracciolo, L., Luca, a. De, & Iannitti, S. (1999). Trajectory tracking control of a four-wheel differentially driven\nmobile robot. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. https://doi.org/10.1109/ROBOT.1999.773994

[17] Lin, L.-C., & Shih, H.-Y. (2013). Modeling and Adaptive Control of an Omni-Mecanum-Wheeled Robot. *Intelligent Control and Automation*, *4*, 166–179. https://doi.org/10.4236/ica.2013.42021

[18] Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics (intelligent robotics and autonomous agents series)*. *Intelligent robotics and autonomous agents*,. https://doi.org/10.1145/504729.504754

[19] Kröse, B. J. A. (1995). Learning from delayed rewards. *Robotics and Autonomous Systems*. https://doi.org/10.1016/0921-8890(95)00026-C

[20] Wang, Y., & Chirikjian, G. S. (2000). A New Potential Field Method for Robot Path Planning. *Proceedings - IEEE International Conference on Robotics and Automation*, *2*(April), 977–982. https://doi.org/10.1109/ROBOT.2000.844727