Collision avoidance algorithm with performance optimization and speed control for multi-robot autonomous system

Ismaiel Ahmed
Ural Federal University (Yekaterinburg, Russia)
Electrical Engineering in Assiut University (Assiut, Egypt)
en.ismaiel@gmail.com

Mikhail Yu. Filimonov
Krasovskii Institute of Mathematics and Mechanics (Yekaterinburg, Russia)
Ural Federal University (Yekaterinburg, Russia)
fmy@imm.uran.ru

Abstract

In this paper a collision avoidance algorithm with speed control and self-optimization in order to improve the performance of the autonomous system is presented. The system consists of a group of mobile robots which move autonomously in a certain area where static obstacles are inserted. In this paper a new term called occupancy ratio is proposed with using collision density to evaluate the performance of the autonomous system. Each robot has a start configuration point and destination configuration and it should move to its destination without any collisions with static obstacles or other moving robots. Simulation was carried out with using MATLAB 2016a. It has found out that the collision density can reflect the performance of the system, the less collision density corresponds to a better performance.

1 Introduction

The main objectives are to move multiple robots in a common workspace to perform a certain mission that demand that each robot find its way from start configuration to a given goal configuration without mutual collision and collisions with obstacles. Often, movement planner approaches are characterized as coupled (centralized) or decoupled planner. A coupled planner deals with all robots as a single combined robot and computes a path in a combined configuration space, while decoupled planner computes a path for each robot independently. In this approach robots find their paths and avoid collisions with obstacles or with each other. Each robot has a sensing scope and can exchange configurations information only with other robots within its sensing scope. Autonomous navigation has been widely investigated and many techniques and approaches were presented. One of effective techniques is a leader-followers conception is proposed in [1] which is used with differential wheels multiple

In: A.A. Makhnev, S.F. Pravdin (eds.): Proceedings of the International Youth School-conference ¡¡SoProMat-2017¿¿, Yekaterinburg, Russia, 06-Feb-2017, published at http://ceur-ws.org

robots system by the mean that according to the leader position, the followers will behave. The results show that probabilistic fuzzy approach with Adaptive Neuro-Fuzzy Inference System (ANFIS) [2] make the system more robustly and improve the performance of the navigation system. Fuzzy perception also is used to collision avoidance in [3] where a non-holonomic robot is used with kinematic and dynamic constraints.

In the paper in section 2 we will discuss the path planning algorithms which we propose for a system of moving robots, in section 3 we will describe the system, taking into account the kinematics characteristics of the robot, the different scopes of the robot, and the equations for the motion behavior, consider the collision avoidance algorithm, the cooperative behavior and sequence of the system. Section 4 shows the results of the simulation.

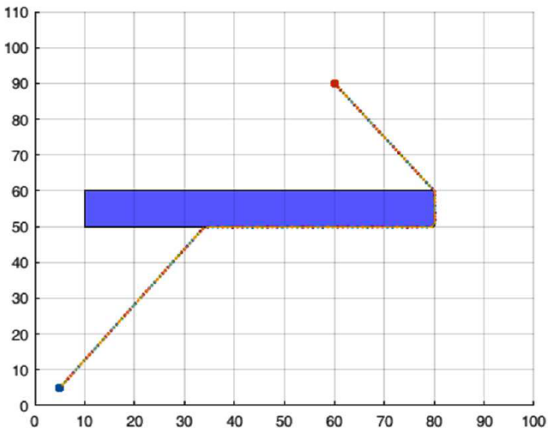

Figure 1: The trajectory calculated by BUG.

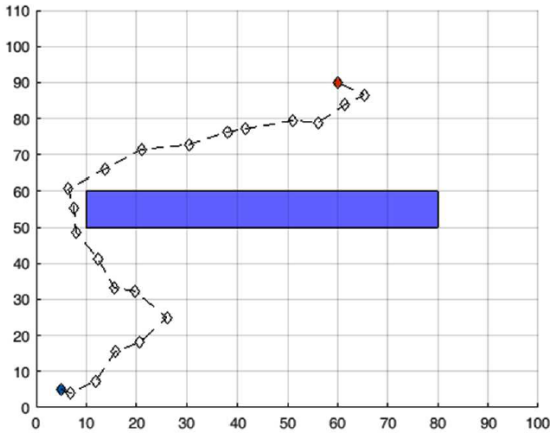

Figure 2: The trajectory calculated by PRM.

2 Path Planning

Path planning problem is how to find a continuous motion that will take a robot from a given initial configuration to a desired destination configuration, subject to the constraint that a robot does not collide with any obstacle in its workspace at any point in the motion. Different approaches have been introduced to implement path planning for mobile robots. Approaches possible to be in accordance with the environment, type of sensor, robot capabilities, etc. Path-planning is needed between start and end point to create a path that should be free of collision and satisfy certain criteria such as shortest path [4].

In our model the robots use Bug and PRM algorithms to a find path to the goal configuration, and it has minimum distance that can sense the environment around it, once obstacle or another robot appear to collide, robots start exchange information about their current position and next position to achieve their goal configuration without collisions.

2.1 PRM Algorithm

Probabilistic roadmaps (PRM) are an effective tool to capture the connectivity of a robot's collision-frees pace and to solve path-planning problems with many degrees of freedom [5]. PRM is used when we have full knowledge of the configuration space of the robot, as it take random samples from this space and testing them, whether they are in a free space then use local planner to connect these samples. After this, the start and goal configurations are added and graph search algorithm (A* Algorithm) is often applied to determine a path between the start and goal configurations.

2.2 Bug Algorithm

Bug algorithm is a local path planning algorithm. Bug algorithms use sensors to detect the nearest obstacle as a mobile robot moves towards a target with limited information about the environment [6]. The algorithm uses obstacle border as guidance toward the target as the robot circumnavigates the obstacle till it finds certain condition to fulfill the algorithm criteria to leave the obstacle toward target point.

2.3 Trajectory Data

Path planning algorithm (PRM or Bug) determines a full trajectory to the robot from the initial point to the goal point, as mentioned before, PRM algorithm is for known environment but Bug is a discovering algorithm for unknown environment. In both cases path-planning divides trajectory into smaller parts of linear trajectories, robot records the coordinates of the points of the start and end of each small trajectory. Such trajectories are free of collisions in terms of static obstacles, for dynamic obstacles we use collision avoidance algorithm. Fig.1 and Fig.2 show the trajectory calculated by Bug and PRM algorithms, respectively, in order to find collision free path to goal configuration and how it's decremented into smaller linear trajectories.

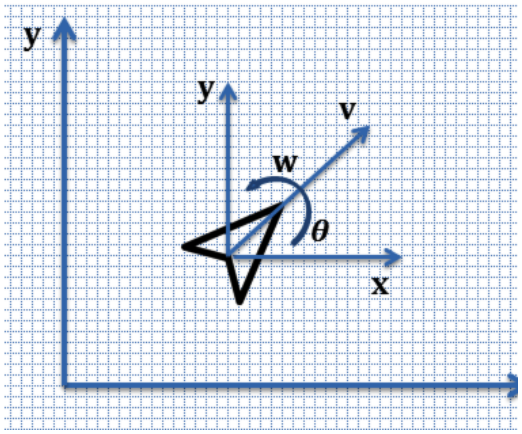

Figure 3: A kinematics model of mobile robot.

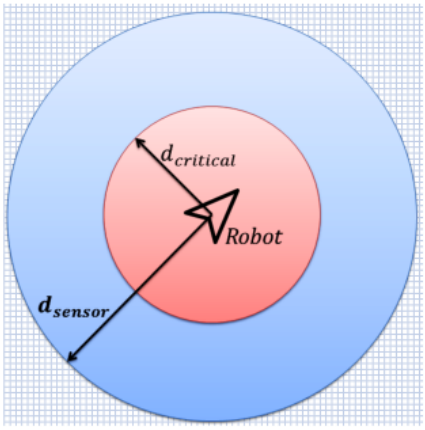

Figure 4: A circular sensing scope of a robot.

3 System Description

3.1 Kinematics of the Robots

Now, the problem how to compute the path planning to move the robot from the initial point to a target point in the unknown environment has to be discussed. In order to control the behaviors of multi-mobile robots we need kinematic analysis of each robot. It's assumed each robot consist of two wheels in a two dimensional plane and robots move without slipping on a plane. The kinematics model of mobile robot and linear movement illustrated in Fig.3 can be represented in (x, y) Cartesian coordinates by the following:

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} \quad \textit{Kinematics during moving,}$$

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} \quad \textit{Kinematics during turning.}$$

It's also assumed that the robots move according to constant predetermined acceleration (positive or negative) till they reach maximum velocity and then move in constant speed ($v = \text{const}$) or till they stop ($v = 0$). Also, as kinematics constraints for robots movement, they can only move in straight lines, and in order to change their directions they need first to slow down to zero velocity then start to turning to the new direction with predetermined angular velocity (ω). Fig.5 shows robots dynamic displacement (distance) with time (period). Robots move in three main modes:

1) acceleration mode,
2) constant velocity mode,
3) deceleration mode.

3.2 Sensing Scope

We assume that the robots are provided by sensors that enable them to discover all objects (moving robots or static obstacles) in a circular sensing scope within a radius of d_{sensor} as shown in Fig.5, also they can find distances between them and any object existed in a distance equal or less than d_{sensor}. Critical distance $d_{critical}$ indicates the distance at which the robot must start to slow down till stand still if any other robot or obstacles are in a distance less than this critical distance, robot should stop and collision avoidance algorithm will be activated.

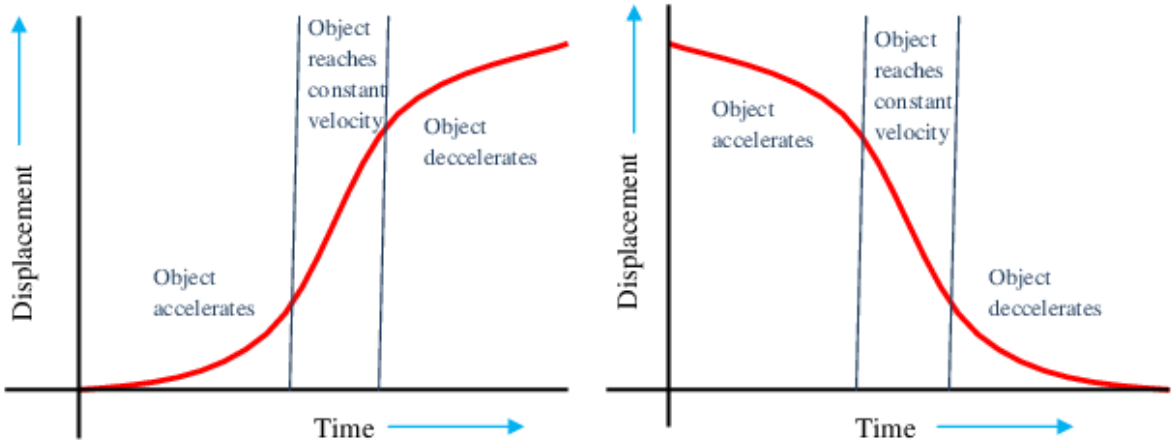

Figure 5: Robots dynamic displacement (distance) with time (period).

3.3 Motion scenario

A robot moves toward its destination coordinates in iterations, at each iteration, the robots calculate the next position according to path planning sequence information. Sensor scope check for expected collisions, it returns a value of (1) if there are any object within a distance less than $d_{critical}$ or value of (0) if there are no objects located in a distance less than $d_{critical}$ radius scope. In the case of (0) feedback, it means that next position is clear of any possible collisions and robot start to continue moving to this position, next position coordinates are being calculated by

$$\textit{Kinematics during turning} \quad \begin{cases} x_{next} = x_{current} + v_1 \cos\theta, \\ y_{next} = y_{current} + v_1 \sin\theta, \\ v_1 = \min\{v + acc, v_{\max}\}. \end{cases} \tag{1}$$

Here, v is instantaneous velocity, acc is acceleration value, θ is angle between x-axis and destination point (x_{dest}, y_{dest}) calculated with

$$\theta = \arctan(y - y_{dest})/(x - x_{dest}). \tag{2}$$

After that linear distance between robot's position and destination point is calculated to determine if the robot reaches its destination and ready to receive a new destination coordinates or it stills not yet reaches its destination and have to repeat the previous calculation.

$$distance = \sqrt{(x_{dest} - x_{current})^2 + (y_{dest} - y_{current})^2}. \tag{3}$$

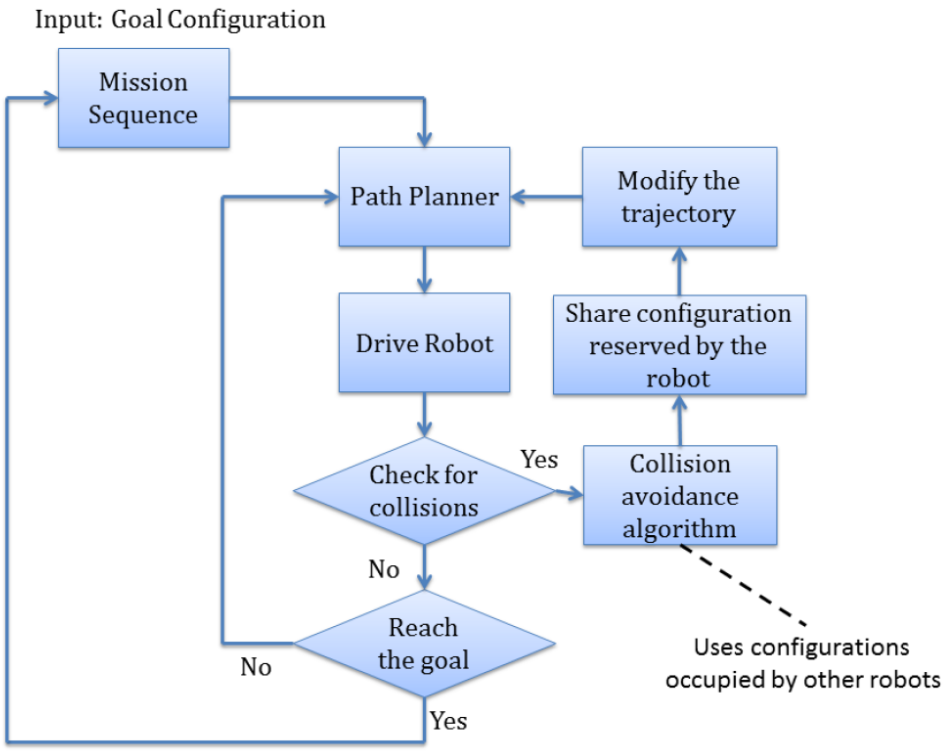

Figure 6: The sequence and interaction between all system corners.

3.4 Collision Avoidance Algorithm

If the sensor data returns value of (1), it indicates the presence of obstacle or moving robots that block robot's path to the destination point and there is a risk of collision, hence the obstacle avoidance algorithm is activated. This algorithm works in the following steps:

1. Slow down the moving robot till zero velocity according to its kinematics restrictions.
2. Start searching for a new free of collisions position according to sensor feedback data.
3. If the distance between the new position and the blocking objects is more than $d_{critical}$, robot's path is modified and robot move to this new position.
4. If the distance is equal or less than $d_{critical}$, position reserved as occupied and algorithm starts searching for another point. The searching for a new position achieved by checking positions on the circumference of distance equal to the critical distance, in each iteration algorithm adds radial interval ($\Delta\theta$) to the default next position to get new possible positions, the sequence of searching for alternative points indicated by numbers in Fig.7. Then algorithm checks if the point is valid or not, this is achieved by the following sequence (n is a number of robots and obstacles in sensing scope).

Step 1. *Checking for the next position of the robot* (x_{next}, y_{next}).

$$d_i = \sqrt{(x_i - x_{next})^2 + (y_i - y_{next})^2}, \quad i = 1, ..., n.$$

$$if \begin{cases} (d_i > d_{critical}), & (x_{next}, y_{next}) = accepted, \\ (d_i \leq d_{critical}), & (x_{next}, y_{next}) = refused. \end{cases}$$

Step 2. *The next position is refused. Adding* $(\Delta\theta)$ *to the current theta of the robot:*

$$(\theta_{(new-r)} = \theta + \Delta\theta) And (\theta_{(new-l)} = \theta - \Delta\theta).$$

Step 3. *Find new coordination of the new two positions for* $(\theta_{(new-r)}$ *and* $\theta_{(new-l)})$ *as in equation* (1).

Step 4. *Check validity of those two positions as in Step 1, next position of the robot is set to be the accepted position, if both positions are accepted, the nearest one to the destination is set to be the accepted position and its coordinates is set as* (x_{next}, y_{next}).

Step 5. *Go to Step 1 (loop).*

This algorithm continues in iterating till it can find positions free of collisions otherwise robot stops and repeat these calculations after time interval Δt.

3.5 Cooperative Behavior

In this model the robots can cooperate and exchange information with each other. For more reality we made some restrictions for this cooperative behavior. First, the robots can only cooperate with other robots located in their sensing scope. Robots can't cooperate with other robots which are located in a distance more than their sensing scope. Second, nearby robots can only interact with limited information, they can only exchange their position coordinates with each other. When robots block the way of each other, collision avoidance algorithm start to search for an alternative free of collision configuration, and share the resulting coordinates with all robots within their sensing scope.

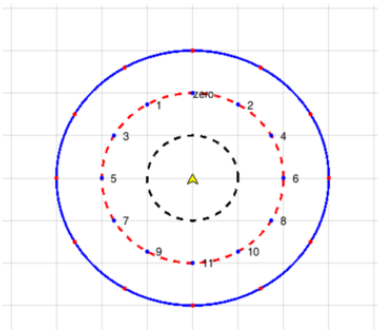

Figure 7: The main three scopes for the robot in shape of three circles.

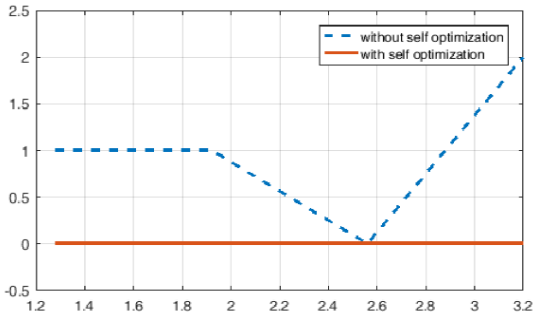

Figure 8: The number of collisions between robots with and without optimization.

3.6 System Sequence Algorithm

Fig.6 shows the sequence and interaction between all parts of the system. Mission consists of many paths and conditions that determine if the mission completed or not, missions are stored in Mission sequence controller which send start and goal configurations to path planner algorithm, this start to find suitable path point sequence that used to guide and drive the robot. At each step sensor checking for collision and according to sensor data collision avoidance algorithm is activated or not, this sequence is repeated till robot reaches its goal and waits for a new mission.

4 Simulation Results

In this simulation using MATLAB R2016a, we simulate the dynamics of a group of robots (2 till 5) moving autonomously through start and goal configurations, which chosen randomly using collisions avoidance algorithms then calculate collision density as a number of collisions per minute. Then each robot configures its own velocity and accelerating parameters to reduce the number of collisions and raise the efficiency of the autonomous system.

Simulation Parameters.

Robot relative dimensions are 2.4 cm x 2.4 cm, area occupied by each robot is 5.76 cm^2, dimension of the configuration space is 30 cm x 30 cm, occupancy ratio is the percentage of space occupied by robots and obstacles to the total area of the configuration space, so the occupancy ratio of 3 robots autonomously moving without the any obstacles is (3 x 5.76 cm^2)/(900 cm^2)=1.92% , acc=0.01 cm/sec^2, $v_{\max}$=0.1 cm/sec, ω=0.1 rad/sec, and the sensor scope $d_{critical}$=5 cm (the critical distance that each robot start to action to avoid collision while any other object are near by less than this distance).

Optimizing process

Robots record its maximum sensing scope d_{sensor} and automatically optimize its maximum velocity $v_{\max}$ and critical distance $d_{critical}$ according to the braking distance which is the distance that robot needs to reach the standstill. $d_{brake}=(v_f^2 - v_{\max}^2)/2a)$, v_f is a final velocity and equals to zero, $v_{\max}$ is an initial velocity, a is a

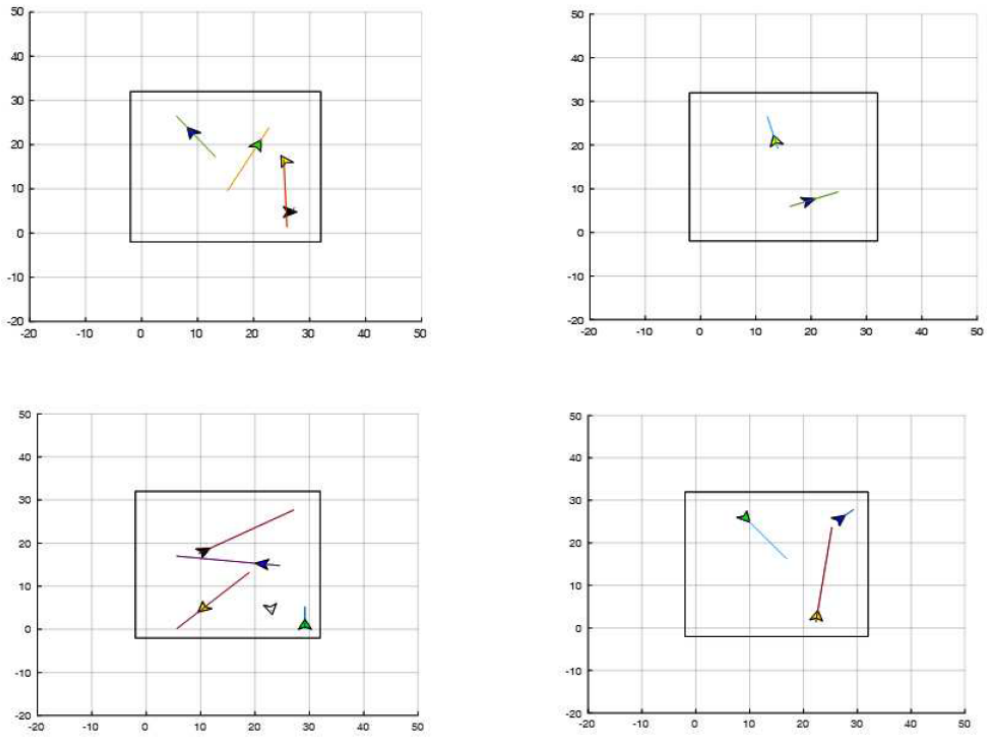

Figure 9: The snapshots of the system simulation model for multi-robot system.

negative acceleration, d_{brake} is a braking distance and in this case $d_{critical} \geq 2d_{brake}$ so that robot can calculate $v_{\max}$ which ensure that robot will stop safely without any object that may suddenly appears in its sensing scope. Fig. 7 shows the main three scopes for the robot in shape of three circles. Robot is in the center. The outer circle represents the sensing scope d_{sensor} of the robot. Robot can be aware of any object exist in the range of this circle and measure the distance between its position and the object's position. The middle circle represents the critical distance $d_{critical}$, any object in the range of this circle represent a danger of collision with the robot. In such case robot starts to brake. The inner circle represents the brake distance d_{brake} that the robot needs to reach zero velocity.

Fig. 8 and Table 1 show the occupancy ratio and collision density. Collision density can be integrated with other factors like duration to evaluate the autonomous systems. It's clear that optimizing the velocity of robots enhance the performance of the system by decrease collision density to zero value. Fig.9 shows snapshots of the system simulation model for multi-robot system on MATLAB.

Table 1: Results. Number of collisions.

No. of Robots	Occupancy ratio (%)	Collision density (no. of collision /min)	
		Robot without optimization	Robots with optimization
2	1.28	0.33	0
3	1.92	0.33	0
4	2.56	0	0
5	3.2	0.67	0

5 Conclusion

In this paper simulation of autonomous system for a group of decoupled multi robots is presented, this simulation take in mind the dynamics restrictions of mobile robots or a group of vehicles. An approach to evaluate the performance of autonomous system by using occupancy ratio and collision density is proposed. Each robot in

this system can optimize its own velocity, search for free configurations and in the same time it collaborates with nearby robots to avoid collisions and achieve the demanded mission. This system is tested with robots up to group of 5 robots using MATLAB. It has found out that the collision density can reflect the performance of the system, the less collision density corresponds to a better performance while occupancy ration has inverse effect on system performance. In case of optimizing the velocity of the robots it is possible to improve the performance of the system as collision density equal to zero.

References

[1] Rami Al-Jarrah, Aamir Shahzad and Hubert Roth. Path Planning and Motion Coordination for Multi-Robots System Using Probabilistic Neuro-Fuzzy. *IFAC-PapersOnline*, 48(10):46-51, 2015.

[2] J.-S. R. Jang. ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems Man and Cybernetics*, 23(3):665-685, May/June 1993.

[3] Byoung-Kyun Shim, Won-Jun Hwang. A Study on Real-Time Implementation of Obstacle Avoidance for Autonomous Travelling Robot. *12th International Conference on Control, Automation and Systems, Jeju Island, Korea*, 2012.

[4] N. Buniyamin, W.A.J. Wan Ngah, N. Sariff, Z. Mohamad. A Simple Local Path Planning Algorithm for Autonomous Mobile Robots, *International Journal of Systems Applications, Engineering & Development*, 2(5):151-159, 2011.

[5] L.E. Kavraki, P. Svestka, J.-C. Latombe, M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566-580, 1996.

[6] V.J. Lumelski, A.A. Stepanov. Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment. *IEEE Transactions On Automatic Control*, 11:1057-1063, 1986.