Proceedings of the 38th Chinese Control Conference
July 27-30, 2019, Guangzhou, China

A Hybrid Reactive Navigation Strategy for a Non-holonomic Mobile Robot in Cluttered Environments

Jian Zhang[1]

1. School of Electrical Engineering and Telecommunications, The University of New South Wales, Sydney 2052, Australia
E-mail: jian.zhang4@unsw.edu.au

Abstract: The reactive collision-free navigation problems are challenging due to the limitation of the environment information. In this paper, we propose a novel hybrid reactive navigation strategy for non-holonomic mobile robots in cluttered environments. Our strategy combines both reactive navigation and Q-learning method. We intend to keep the good characteristics of reactive navigation algorithm and Q-learning and overcome the shortcomings of only relying on one of them. The performance of the proposed strategy is verified by computer simulations, and good results are obtained.

Key Words: Obstacle avoidance, reactive navigation, cluttered environments, shortest path planning, Q-learning, reinforcement learning

1 Introduction

As the demands of autonomous mobile robots are increasing in recent years, the collision-free navigation problems draw more attention in robotics field. In order to reach the target in a complex environment without collisions, the robot should enable to detect and avoid obstacles. The existing navigation approaches can be classified into two main categories: global and reactive (local) navigation approaches [1]. In the global navigation algorithms, *a priori* information is known, and will be used to build the model of the environment (i.e. "map") and try to find out the best possible solution [2–4]. On the other side, reactive navigation algorithms use mounted sensors to observe small fraction of the unknown environment at each time [3, 5, 6]. Compared with the global navigation methods, the reactive navigation algorithms have less computation load and are more practical, as no *a priori* information of the environment needed.

Although the navigation algorithms show the ability to maneuver the robot in an unknown environment with good robustness, and enable to reach a steady target without any collisions, they cannot make sure the optimality of the trajectory. In other words, they cannot guarantee to find out the best possible trajectory/path (shortest in the length) in each operation, due to the limitation of environment information. On the other hand, in those complex environment that is difficult to model, the navigation algorithm with reinforcement learning methods [7] may be applicable to plan the best possible path for the robot, but need a huge numbers of trails for training [8–12]. Another severe limitation is that many other researchers' papers [2, 11–13] ignore the non-holonomic constrains of the moving robots, which cause severe limitation in practice.

The shortcomings mentioned above motivated researchers to propose hybrid approaches by combing reactive navigation and reinforcement learning methods to accomplish better navigation results[14]. However, several issues need further investigation. In [15], a reinforcement learning method with the concept of sliding mode control is proposed, and the researchers proved their new control method has advantages of both the robustness and stability of the sliding mode control and applicability to complex systems that are hard to model. It is still a long way to apply this method to the real navigation environment. Another possible approach to solve the navigation problem is to modify the traditional reinforcement learning algorithm, such as [11], which developed a quantum-inspired reinforcement learning algorithm to improve the robustness. The applicability in practice and the training time cannot be settled, which is a huge problem in practice[16, 17].

In this paper, we propose a novel hybrid reactive collision-free navigation algorithm under an unknown environment cluttered with random form of smooth (possibly non-convex) obstacles with constraint on the curvature of their boundaries. And the simulation results confirm the validity of our algorithm and show that the robot is able to seek the shortest path to reach the steady target with only several trails. We assume that the ground robot studied in this paper is a unicycle, which can be considered as a Dubins car [18] with non-holonomic constraint. Compared with omnidirectional mobile robot model [2, 11, 13], the non-holonomic model is more practical. This model has been widely used to describe the movements of different mechanical systems such as wheeled robots, unmanned aerial vehicles, and missiles and references therein [3, 19–21].

The remainder of this paper is organised as follows. Section 2 formulates the problem we are going to solve. The navigation policies and Q-learning based path planning algorithm are presented in Section 3 and 4 respectively. The computer simulations are performed to validate the performance of the presented approach, and the results can be found in section 5. In these simulations, two scenarios were considered with random-shaped obstacles. Section 6 includes brief conclusions.

2 Problem Formulation

We consider a planar robot modelled as a unicycle, and we can represent its state as the configuration $X = (x, y, \theta) \in SE(2)$, where $(x, y) \in \mathbb{R}^2$ is the robot's Cartesian coordinates, and θ is the robot's heading. The robot travels with speed v and angular velocity u, both constrained by given constants. The dynamics of the robot can be represented as

This work was supported by the Australian Research Council.

SE(2): the special Euclidean group in the plane

follows:

$$\begin{aligned}\dot{x}(t) &= v(t)\cos\theta(t)\\ \dot{y}(t) &= v(t)\sin\theta(t)\\ \dot{\theta}(t) &= u(t)\end{aligned} \tag{1}$$

where

$$u(t) \in [-u_M, u_M], v(t) \in (0, v] \tag{2}$$

Where u_M is the maximum angular velocity, the non-holonomic constraint can be described as

$$|u(t)| \leqslant u_M \tag{3}$$

And the minimum turning radius of the robot satisfies

$$R_{min} = \frac{v}{u_M} \tag{4}$$

Any path $(x(t), y(t))$ of the robot as shown in (1) is a plane curve satisfying the following constraint on its so called average curvature (see [3]): let $P(s)$ be this path parametrised by arc length, then

$$\|P'(s1) - P'(s2)\| \leqslant \frac{1}{R_{min}}|s1 - s2| \tag{5}$$

Here $\|\cdot\|$ denotes the standard Euclidean vector norm. Note that we use the constraint (4) on average curvature because we cannot use the standard definition of curvature from differential geometry [22] since the curvature may not exist at some points of the path of the robot.

This unicycle model is moving in an unknown planar area, with several disjoint steady obstacles, $D_1, ..., D_k$, the safety margin of obstacle is given as $d_{safe} > 0$, and then the safety boundary of the obstacle $\partial D_i(d_{safe})$ can be found, see Fig. 1(a). For the challenge of reactive navigation in unknown environment crowded by numbers of possibly non-convex obstacles with constraint on the curvature of their boundaries, the shortest path consists of edges of the so-called tangent graph is proven in [2].

Assumption 2.1. *Any two elements do not intersect.*

The tangent line can be defined as a straight line that tangent to two elements simultaneously and do not intersect with each other. The points of elements belonging to tangent lines are called tangent points. And the curve between two tangent points on the same element is called arc. The tangent graph, denoted as $G(V, E)$, where vertex set V consists of finite set of tangent points and edge set E consists of finite set of arcs and tangent lines. Fig. 1(b) shows an example of a tangent graph. Therefore, the trajectory of the robot at different position can be described easier.

Definition 2.1. *There are two circles with the radius R_{min} that cross the initial robot position $(x(0), y(0))$ and tangent to the robot initial heading $\theta(0)$. We will call them initial circles.*

Assumption 2.2. *For all i, the boundary ∂D_i of the obstacle D_i is a closed, non-self-intersecting analytic curve.*

Assumption 2.3. *For all i, the boundary $\partial D_i(d_{safe})$ is a closed, non-self-intersecting analytic curve with curvature $k_i(p)$ at any point p satisfying $k_i(p) \leqslant \frac{1}{R_{min}}$.*

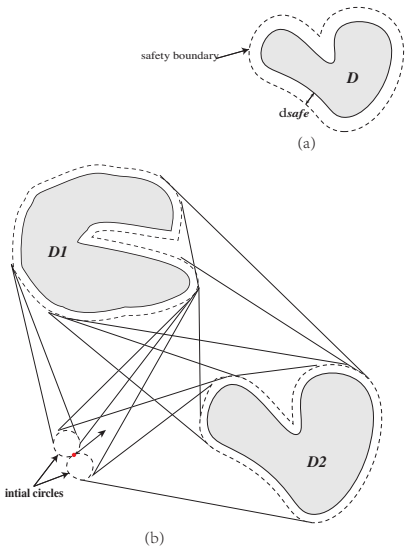


Fig. 1: An illustrative example of (a) safety margin (b) tangent graph

Assumption 2.4. *The minimum distance between obstacles is big enough for robot to travel safely.*

Assumption 2.5. *The robot's location $(x(t), y(t))$ and heading $\theta(t)$ are obtained by some localization technologies, like odometry, GPS, etc.*

3 Navigation Policies

In this section, we consider the case when the robot does not know the location of the target or the obstacles *a priori*. The robot has several ultrasonic-type sensors around it, which help the robot to detect its current distance between the target and obstacles, and odometer to estimate its position. So that the robot is able to determine the relative coordinates of the targets and the boundaries of obstacles.

We will need the following assumptions and definitions to simplify the case.

Assumption 3.1. *Each tangent point only belongs to one tangent line.*

Definition 3.1. *When the robot is moving along the obstacle boundary or one initial circle and reaching a tangent point. A robot is defined to reach an exit tangent point, when it leaves its current arcs and heads to the corresponding tangent line between two elements of the tangent graph if the robot's orientation is the same as the direction of this corresponding straight line.*

According to [3], we approach this problem by reasoning over policies:

$$\xi = \{initial, pursuit, follow\} \tag{6}$$

Where *initial* refer to the policy that the initial condition of the robot is move along one of these two initial circles randomly. When the robot moves along the initial circle or obstacle boundary and reach an exit tangent point to the target, it begins to operate *pursuit* policy to move along the corresponding tangent line between its current tangent point and the target. Moreover, the robot will use the path planning algorithm in next section to decide between *pursuit* and *follow* policies, which means move to corresponding tangent line and follow current initial circle or obstacle boundary

arc, respectively. Otherwise, the robot operates *follow* when it moves along a corresponding tangent line between initial circle and obstacle or between two obstacles, and reaches the tangent point on the obstacle boundary.

This navigation strategy was realised as a sliding mode control law by switching between the boundary following approach and the pure pursuit navigation approach from [3] as follows:

$$u(t) = \begin{cases} \pm\, u_M & R1 \\ \Gamma sgn[\phi \tan(t)] u_M & R2 \\ \Gamma sgn[\dot{d}_{min} + \chi(d_{min}(t) - d_{safe})] u_M & R3 \end{cases} \quad (7)$$

Where R1 refers to *initial* mode, R2 refers to *pursuit* mode, and R3 refers to *follow* mode. The variable $\Gamma = +1$ when robot's heading direction intersects with obstacles, and $\Gamma = -1$ otherwise.

The linear saturation function is defined as follows:

$$\chi(r) = \begin{cases} lr & if\ \|r\| \leqslant \delta \\ \delta l sgn(r) & otherwise \end{cases} \quad (8)$$

Where $sgn(r) := 1$, when $r > 0$, $sgn(r) := 0$, when $r = 0$, and $sgn(r) := -1$, whereas $r < 0$. $l, \delta > 0$ are tunable constants.

And the navigation mode transition rules can be illustrated as:

$$\begin{aligned} R1 \rightarrow\ R2: &\quad reach\ the\ tangent\ point \\ &\quad to\ target\ or\ obstacle \\ R2 \rightarrow\ R3: &\quad d_{min}(t) < d_{trig}, \dot{d}_{min}(t) < 0 \\ R3 \rightarrow\ R2: &\quad illustrate\ in\ next\ section \end{aligned} \quad (9)$$

According to Eq. (7), the navigation is fulfilled by switching between modes R1 - R3. At the beginning of each episode, R1 is active, and transits to other modes are determined by Eq. (9). Mode R1 describes the robot motion that move along the initial circle with maximal actuation. Mode R2 describes the robot motion that pursuit along the tangent segment, where $\phi \tan(t)$ is defined as the angle between the vehicle's heading and a line segment connecting the vehicle and currently tracked tangent edge. Mode R3 describes boundary following behaviour, where the control calculation is based on the minimum distance to the nearest obstacle, defined as $d_{min}(t)$. A constant $d_{trig} > d_{min}(t)$ is also introduced, which determines when the control system transitions to boundary following mode. In the following section, we will describe the procedure to transit from R3 to R2. We introduce Q-learning to find the best possible p for the certain environment.

4 Path Planning Algorithm

In this paper, we apply Q-learning [23] to find the best possible p at each exit tangent point. The reinforcement learning is a real-time, online learning method [8]. By introducing Q-learning, the robot is able to sense more about the environment and learn to choose the optimal actions to achieve the goal, in our paper, reach the target. The strategy that the robot learns from the environment and perform properly is by matrices and the numerical evaluation function, which assigns numerical values to different distinct actions at each distinct state. The policy of the robot's learning is to achieve as high reward scores as it can with long-term interest. The robot will be reward by"good" behaviour and punished by "bad" one. In our case, the robot can perceive the distinct state as a set of $\mathcal{S}$, $\forall s \in \mathcal{S}$, and its actions as a set of $\mathcal{A}$, $\forall a \in \mathcal{A}$ at each discrete time step t, respectively. At each discrete time step t, the robot senses the current state s_t, chooses a current action a_t and performs it, the environment responds by returning a reward $R(s_t, a_t)$ and by producing the immediate successor state value $Q(s_t, a_t)$, who depend only on the current state and action. In other words, the robot regularly updates its achieved rewards based on the taken action at a given state. The schematic of Q-learning shows in Fig. 2.

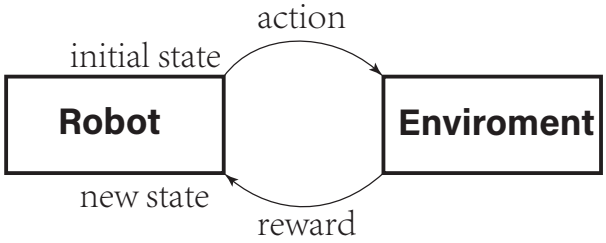


Fig. 2: Schematic of Q-learning

As it is impossible to predict in advance the exact outcome of applying an arbitrary action to an arbitrary state, we apply $Q(s_{t+1}, a_{t+1})$ (i.e. the Q-value) of the robot, which indicates the maximum discounted reward that can be achieved starting from S and applying action A first. With the action a_t (moving along the boundary or the corresponding segment) at a given state s_t (current exit tangent point's relative coordinate), the future total reward $Q(s_{t+1}, a_{t+1})$ of the robot is computed using

$$\begin{aligned} Q(s_{t+1}, a_{t+1}) =& (1-\alpha) Q(s_t, a_t) + \alpha [R(s_t, a_t) \\ &+ \gamma Q_{max}(s_t, a_t)] \end{aligned} \quad (10)$$

where $R(s_t, a_t)$ is the immediate reward of performing an action a_t at a given state , $Q_{max}(s_t, a_t)$ is the maximum Q-Valve and α is the learning rate, which is between 0 and 1, and $\gamma \in [0, 1)$, which is the discount factor. By following an arbitrary policy that produces the greatest possible cumulative reward over time, there exits the cumulative discounted reward G_t achieved from an arbitrary initial state s_t as follows:

$$\begin{aligned} G_t =& R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \\ =& \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \end{aligned} \quad (11)$$

The discount factor γ determines the relative value of delayed versus immediate rewards, R_{t+k+1} are rewards of successor actions generated by repeatedly using the policy.

From the above algorithms, we can make the robot more concentrate on the immediate reward or the previous experience by increasing α or γ and vice versa. In other word,

Table 1: Path Planning Algorithm

Algorithm 1 Path Planning Algorithm

Set the γ parameter and environment rewards matrix R
Initialize matrix $Q(s_t, a_t)$ to zero
for each episode **do**
    Select a random state s_t from $\mathcal{S}$
    while the goal state hasn't been reached **do**
        Choose one possible action a_t from current state s_t by matrix R
        Consider moving to the next possible state s_{t+1}
        Based on all possible actions, computer the maximum Q value for s_{t+1} from current matrix Q
        Refresh matrix Q by updating the value of $Q(s_t, a_t)$ derives from $Q(s_t, a_t) = R(s_t, a_t) + \gamma Q_{max}(s_t, a_t)$
        Set S_{t+1} as the current state
    end while
end for

larger α will make the robot more concern about the immediate reward, whereas larger γ will lead to actions which pays more attention to previous experience, see Fig. 3 for an example.

The training can be finished by numbers of training episodes. For each episode, it contains series of observations, actions, and rewards at each state from the initial observation to the terminal observation. The robot selects one among all possible actions for the current state, using this possible action to consider going to the next state. Furthermore, the robot tries to get maximum Q value for this next state based on all possible actions by Eq. (10), upload the new Q value to the Q-table (i.e. a large table with separate entry for each distinct state-action pair) and set the next state as the current state. The robot is going to iterate the above learning process until the goal state has been reached, which is the end of one episode. Provided that the system can be modelled as a deterministic Markov Decision Process (MDP), which means for a distinct state, the output is a certain action, the training episodes will be iterated until the robot's estimate Q table converges to the actual Q, and actions are chosen so that every state-action pair is visited infinitely often.

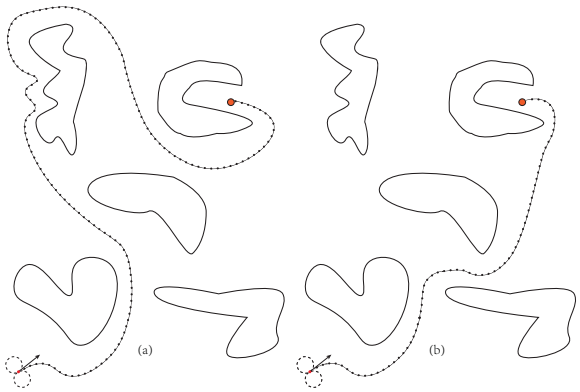


Fig. 3: (a) One random trajectory generated without proposed algorithm and (b) the navigation trajectory after proposed algorithm's training

5 Simulation Results

Table 2: Simulation parameters

Parameter	Value
Maximum angular velocity u_M	60 degree/s
Maximum velocity v	0.5 m/s
Safety margin d_{safe}	$>$ 0m
Tunable constant l	0.08
Tunable constant δ	0.5
Transition margin d_{trig}	0.8 m
Discount factor γ	0.8
ε	0.4
range of probability p	[0,1)

To validate the effectiveness of the proposed algorithm, two simulation scenarios are carried out considering different conditions and environments. Firstly, the $25m \times 25m$ unknown environment filled with random-shaped obstacles is considered. The more crowded environment is handled in the second. The performance of proposed algorithm is evaluated by comparing the trajectories with and without the algorithm.

The control law was simulated using the perfect discrete time model, updated at a sample time of 0.1s. The control parameters are shown in Table. 2. The robot has no *a priori* information about the environment, it can detect its surroundings only by 10 mounted ultrasonic sensors. Simulations were carried out on V-REP, interfaced with MATLAB and Python.

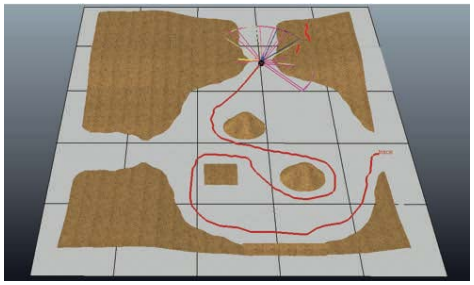
In order to figure out how these two algorithms works, we simulate the first scenario, and second scenarios as follows. Compare with the randomized navigation algorithm, for the robot in our proposed algorithm, the introducing of Q-learning can help robot to make the best decision at each exit tangent point.

A. Unknown cluttered environment with random-shaped obstacles

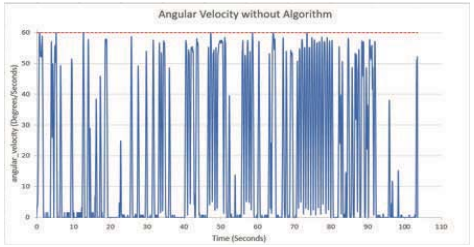
The first scenario aims to validate the proposed navigation algorithm in an unknown cluttered environment with random-shaped obstacles. The situation in Fig. 4 is evaluated. It consists of random-shaped sand dunes as the obstacles. Fig. 4 also shows how the robot plans its path, and its velocity and angular velocity changes with time before and after the proposed algorithm's training, respectively. The absolute value of the angular velocities and the speeds of the robot are within the constrains. And the average value probability p after training is less than its give constrain for this particular cluttered environment. With our simulation results, we found out that the average value of the probability p is susceptible to the unknown environment. As can be seen from Fig. 4, the robot's total length of the trajectory for this environment is significantly deducted due to the application of our algorithm.

B. More challenging cluttered environment with random-shaped obstacles

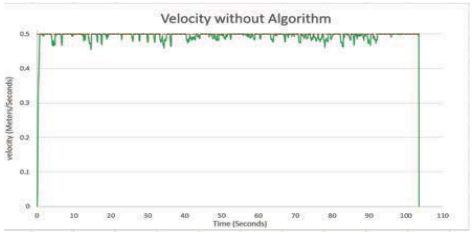
The second scenario intends to discuss the performance of the algorithm with more cluttered deployed obstacles shown in Fig. 5, which means the distance between dunes becomes closer. The simulation results could be attributed to that the proposed algorithm can always find out the shortest path of



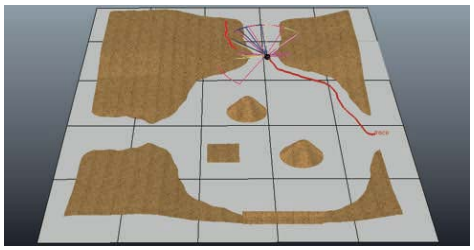
(a) The trajectory of the robot



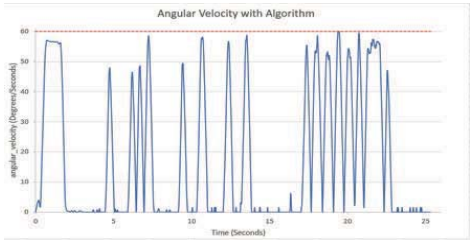
(b) Robot's absolute value of angular velocity



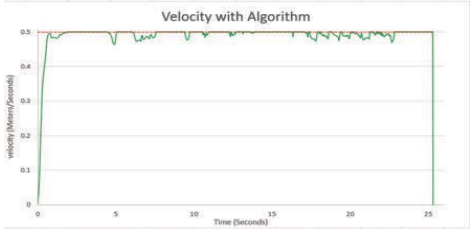
(c) Robot's velocity



(d) The trajectory of the robot



(e) Robot's absolute value of angular velocity



(f) Robot's velocity

Fig. 4: (a), (b), (c) Robot's trajectory, its absolute angular velocity, and velocity respectively, generated without our proposed algorithm, and (d), (e), (f) are robot's trajectory, its absolute angular velocity, and velocity respectively after proposed path planning algorithm's training in an unknown cluttered environment with random-shaped obstacles.

the operating area within a specific time, while the randomized algorithm may spend more time to the target, thus may use longer path. By comparing Fig. 5 (a), (b), (c) with (d), (e), (f) respectively, we can find the advantage of proposed algorithm becomes more evident for the operating area with both first and second scenarios, which is more difficult for the random search to operate. Compared with traditional Q-learning algorithm, our proposed algorithm simplify the complex calculation and react swiftly.

We show the benefits of introducing Q-learning in terms of navigation performance by efficiently finding out the shortest path in an unknown complex steady environment. The merit of our proposed algorithm can be reflected even by simply comparing the length of the path, and the total time of process with the randomized navigation algorithm.

6 Conclusion and Future Research

In this paper, we considered the collision-free reactive navigation problem in the environment cluttered with obstacles constrained on the curvature of their boundaries. We proposed a hybrid reactive navigation algorithm that can be operated in cluttered environments and the results successfully validated that the robot can find the shortest path to the steady target without any collisions. A series of navigation policies are adopted as the fundamental of our collision-free navigation. What's more, a proposed path planning algorithm taught the robot how to switching between these policies by the selections of probability p at each exit tangent point, in order to make sure the robot can find out the shortest path during its reactive navigation. The performance and effectiveness of our algorithm has been confirmed by extensive computer simulations in different scenarios.

An important direction for future research is to extend the obtained results to the case on a team consisting of several robots, for instance, in the problem of navigation for sweep coverage in cluttered environments [24].

References

[1] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.

[2] Y. H. Liu and S. Arimoto, "Path planning using a tangent graph for mobile robots among polygonal and curved obstacles," *International Journal of Robotics Research*, vol. 11, no. 4, pp. 376–382, 1992.

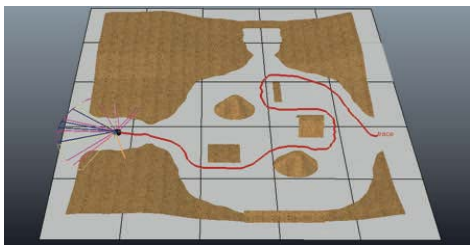
[3] A. V. Savkin and M. Hoy, "Reactive and the shortest path navigation of a wheeled mobile robot in cluttered environments," *Robotica*, vol. 31, no. 2, pp. 323–330, 2013.

[4] C. Wang, A. V. Savkin, and M. Garratt, "A strategy for safe 3D navigation of non-holonomic robots among moving obstacles," *Robotica*, vol. 36, no. 2, pp. 275–297, 2018.

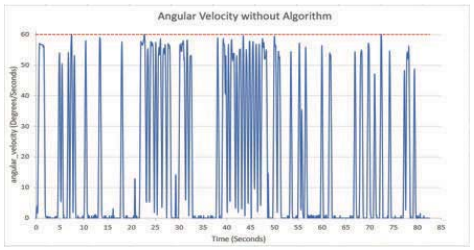
[5] A. V. Savkin and C. Wang, "Seeking a path through the crowd: Robot navigation in unknown dynamic environments with moving obstacles based on an integrated environment representation," *Robotics and Autonomous Systems*, vol. 62, no. 10, pp. 1568–1580, 2014.

[6] A. S. Matveev, M. C. Hoy, and A. V. Savkin, "A globally converging algorithm for reactive robot navigation among moving and deforming obstacles," *Automatica*, vol. 54, pp. 292–304, apr 2015.

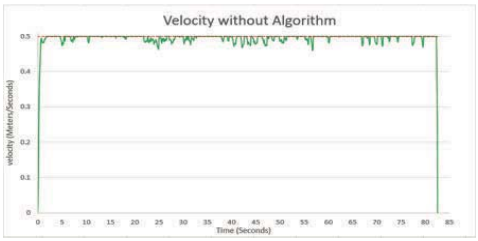
[7] M. W. van Otterlo eds and Martijn, *ALO 12 - Reinforcement Learning*. 2013.



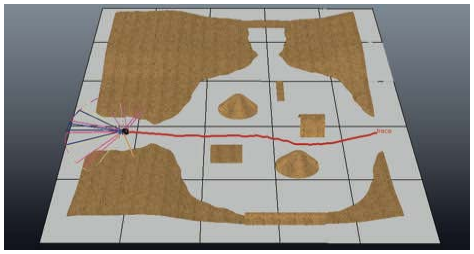
(a) The trajectory of the robot



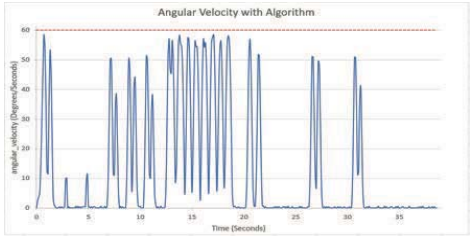
(b) Robot's absolute value of angular velocity



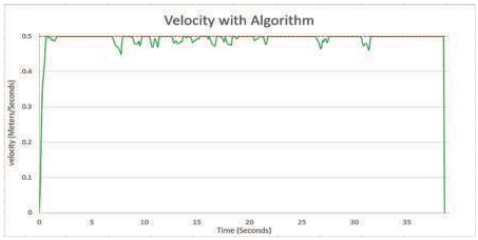
(c) Robot's velocity



(d) The trajectory of the robot



(e) Robot's absolute value of angular velocity



(f) Robot's velocity

Fig. 5: (a), (b), (c) Robot's trajectory, its absolute angular velocity, and velocity respectively, generated without our proposed algorithm, and (d), (e), (f) are robot's trajectory, its absolute angular velocity, and velocity respectively after proposed path planning algorithm's training in a more challenging cluttered environment with random-shaped obstacles.

[8] Q. Zhang, M. Li, X. Wang, and Y. Zhang, "Reinforcement learning in robot path optimization," *Journal of Software*, vol. 7, no. 3, pp. 657–662, 2012.

[9] J. Cai, R. Yu, and L. Cheng, "Autonomous navigation research for mobile robot," in *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, pp. 331–335, IEEE, jul 2012.

[10] J. Cai and L. Li, "Figure 1. Model of operant conditioning learning. Autonomous Navigation Strategy in Mobile Robot," 2013.

[11] D. Dong, C. Chen, J. Chu, and T. J. Tarn, "Robust quantum-inspired reinforcement learning for robot navigation," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 1, pp. 86–97, 2012.

[12] S. Thrun, "An approach to learning mobile robot navigation," *Robotics and Autonomous Systems*, vol. 15, no. 4, pp. 301–319, 1995.

[13] P. J. Costa, N. Moreira, D. Campos, J. Gonçalves, J. Lima, and P. L. Costa, "Localization and Navigation of an Omnidirectional Mobile Robot: The Robot@Factory Case Study," *Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 11, no. 1, pp. 1–9, 2016.

[14] J. Shabbir and T. Anwer, "A Survey of Deep Learning Techniques for Mobile Robot Applications," 2018.

[15] M. Obayashi, N. Nakahara, T. Kuremoto, and K. Kobayashi, "A robust reinforcement learning using the concept of sliding mode control," *Artificial Life and Robotics*, vol. 13, no. 2, pp. 526–530, 2009.

[16] S. G. Khan, G. Herrmann, F. L. Lewis, T. Pipe, and C. Melhuish, "Reinforcement learning and optimal adaptive control: An overview and implementation examples," *Annual Reviews in Control*, vol. 36, no. 1, pp. 42–59, 2012.

[17] M. A. Kareem Jaradat, M. Al-Rousan, and L. Quadan, "Reinforcement based mobile robot navigation in dynamic environment," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 135–149, 2011.

[18] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, vol. 79, p. 497, jul 1957.

[19] A. S. Matveev, H. Teimoori, and A. V. Savkin, "A method for guidance and control of an autonomous vehicle in problems of border patrolling and obstacle avoidance," *Automatica*, vol. 47, no. 3, pp. 515–524, 2011.

[20] A. S. Matveev, A. V. Savkin, M. Hoy, and C. Wang, *Safe robot navigation among moving and steady obstacles*. Elsevier, 2015.

[21] M. Defoort, J. Palos, A. Kokosy, T. Floquet, and W. Perruquetti, "Performance-based reactive navigation for non-holonomic mobile robots," *Robotica*, vol. 27, no. 2, pp. 381–390, 2009.

[22] C. E. Springer and D. J. Struik, "Lectures on Classical Differential Geometry.," *The American Mathematical Monthly*, vol. 59, no. 2, p. 119, 1952.

[23] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[24] T. M. Cheng and A. V. Savkin, "Decentralized control for mobile robotic sensor network self-deployment: Barrier and sweep coverage problems," *Robotica*, vol. 29, no. 2, pp. 283–294, 2011.