Petri Net Based Navigation Planning with Dipole Field and Dynamic Window Approach for Collision Avoidance

Lan Anh Trinh, Mikael Ekström and Baran Cürüklü

***Abstract*— This paper presents a novel path planning system for multiple robots working in an uncontrolled environment in the presence of humans. The approach combines the use of Petri net to plan the movement of multiple robots to prevent the risk of congestion caused by routing several robots into a narrow region, together with a dipole field with dynamic window approach to avoid collisions of a robot with dynamic obstacles. By regarding the velocity and direction of both humans and robots as a source of magnetic dipole moment, the dipole-dipole interaction between the moving objects will generate repulsive forces to prevent collisions. The whole system is presented on robot operating system platform so that its implementation can be extendable into real robots. Experimental results with Gazebo simulator demonstrates the effectiveness of the proposed approach.**

I. INTRODUCTION

Path planning is a core component of a robot for finding an optimal path from a start to a goal with respect to a cost function. The path needs to be collision free, i.e. the robot is able to avoid collisions with static, and moving objects along its path. Although path planning has been addressed thoroughly since 1990s [1, 2], many solutions of the problem has been restricted to a static environment or a single robot. Recently, new challenges have emerged for path planning of (semi-)autonomous robots within the Industry 4.0 context [3]. With an example of an automated warehouse, more advanced features are needed nowadays for robotic agents to deal with changing environments as well as moving objects, that introduces some levels of uncertainty into the path planning problem. In industrial areas, a robot may share the working space with humans and other robots, leading to a requirement of cooperation of multiple robots to address the navigation tasks at a large scale. As autonomous control allows the robot to have a freedom of movement as well as a direct interaction between robot and human, there is a critical need of setting a high safety level of autonomous robots to prevent any dangerous actions to a user. At the same time, a robot must complete the task with correct outputs and within a specified time. Those issues lead to the critical needs of dependable properties for the path planning algorithm [4]. The main focuses of those dependability properties are centered on *availability, reliability*, and *safety* with the aim toward a correct and reliable safe path planning algorithm for multiple robots and to ensure a safe path planning service to avoid any unwanted consequences on humans, other robots, and finally the environment.

*School of Innovation, Design, and Technology, Mälardalen University, Västerås, Sweden anh.lan@mdh.se, mikael.ekstrom@mdh.se and baran.curuklu@mdh.se

Solutions with the aim of dependable path planning start with the understanding of the threats to the system dependability which includes failures, errors and faults. A *failure* happens when the service provided by a system does not comply with its specifications. This results in an *error*, which affects the services, and will/may cause a *fault*. This fault is thus the main root leading to the failure of the system. The implementation of a dependable system focuses on different means such as fault prevention, fault removal, fault analysis, and fault tolerance. Among different approaches, Petri net (PN) provides a broad solution to address the dependability of an autonomous system with regards to analyse and solve conflicts and sharing of resources. Related works of PN for system modelling and analysis are found by Samanta and Sarkar [5]Meanwhile, the uses of PN to design autonomous robots are demonstrated by Yasuda et al. [6], Iocchi et al. [7] and to implement fault tolerance by Miyagi et al. [8], Lusier et al. [9]. Moreover, PN has been used with robot operating system (ROS) to be implemented into real robots by Fabre et al. [10], which was also applied for collaborative robots by Lill et al. [11].

The open challenges for development of dependable path planning of an autonomous robot involves the fact that the robot must be able to recognise and work with humans, and/or other robots, in a unstructured and unknown environment. Currently, in the most widely used robotic framework, ROS, the navigation stack, including global path planning and local path planning with dynamic window approach (DWA) [12], mainly considers humans and other moving objects as static obstacles. This may lead to slow response when the robots are to avoid close obstacles. Besides, the navigation of several robots into one narrow area may make them to block each other and harder to find the way out from the area to reach their goals, e.g dead- or live- lock situations. To address the congestion of routing multiple robots into one place, a group of robots should proactively define effective routes to avoid conflicts with each other. As aforementioned is PN an effective tool in dependable autonomous control to resolve conflicting problems, the first contribution of this paper presents a new approach of using PN to plan the paths of multiple robots with a delay to avoid routing many robots into a same place to avoid congestion. Moreover, to address the other issue with considering moving objects as static obstacles in the conventional local path planning with DWA, the next contribution of this paper presents a new local path planning with dipole fields on top of DWA for obstacle avoidance based on the moving directions and speeds of dynamic objects.

The rest of the paper is organised as follows. The backgrounds of PN model, global path planning and local path planning with DWA are presented in Section 2. The proposed path planning algorithm based on PN planning and obstacle avoidance based on DWA and dipole field are demonstrated in Section 3. Further on Section 4 shows the experiments to evaluate the proposed system. Finally, Section 5 concludes the paper with contributions of the paper and future works.

II. BACKGROUNDS

A. Petri Net (PN) Model

PN is defined in mathematical aspect as a bipartite graph of a set of tuples (P, T, W), where $P = \{p_1, p_2, ..., p_{|P|}\}$ and $T = \{t_1, t_2, ..., t_{|T|}\}$ are disjoint sets of places and transitions, $|P|$ and $|T|$ are the number of elements of P and T respectively, and $W \subseteq (P \times T) \cup (T \times P)$ is a set of arcs connecting from a place to transition and vice versa. The arcs that go out from a place to a transition are named as the input places of transitions. While the ones running out from a transition to place are the output places of transitions. The weights are added to the input and output flows of each transition. With regards to a set of output weights O and a set of input weights I, PN is described as a set of five tuples $G(P, T, W, O, I)$. Places in a PN may consist of a number of marks named tokens. The number of available tokens at place p is represented by $M(p)$, and M is marking vector that denotes availability of tokens at all place $p \in P$. The marking M is expressed as a vector $[M(p_1), M(p_2), ..., M(p_i), ..., M(p_{|P|})]$, in which p_i is a place, $|P|$ is the number of places in PN, and $M(p_i)$ is the number of tokens at the place p_i. Let O be a two dimensional matrix of weights $O(p_i, t_j)$ from the place p_i to the transition t_j. I is similarly defined by the weight $I(t_j, p_i)$ from the transition t_j to the place p_i. It is noted that $1 \leq j \leq |T|$, where $|T|$ is the total number of transition. Thus, a change of the marking vector of a transition from M to M' is given by a finite sequence of transitions

$$M'(p) = M(p) + I(t, p) - O(p, t), \forall p. \tag{1}$$

It is said that the marking M' is reached by the marking M by firing t. In general, the marking M' is reachable from M by a finite sequence of k transitions $\sigma = t_{i_1} t_{i_2} ... t_{i_k}$. With an initial marking M_0, the full description of a PN consists of six tuples $G(P, T, W, I, O, M_0)$. A full graph of all possible markings and transitions, i.e. a reachability set, is described by state-space analysis. As the number of vertices and edges of the state-space graph increase dramatically with regards to the number of places and transitions, the state-space analysis is limited to a small PN.

B. Global Path Planning and Theta* Algorithm

In general, the path planning system of an robot is divided into local path planning and global path planning. The local path planning controls the movements of a robot within a local area once the route of the robot to the goal has been established. Meanwhile, the global path planning needs information regarding the environment e.g. as a scanned map beforehand to generate a global path from a starting point to a desired goal. However, the global path planning mainly applies for a static environment. This means that in order to compute the global path to a goal for the robot, the presence of the other robots and human, i.e. dynamic objects, are not taken into account, but only static obstacles in the working space are concerned. Conventionally, the algorithms used for planning the global paths are graph/map/grid search-based algorithms. The Dijkstra algorithm [1] and its extension A* [2] are well-known approaches for searching in a map. Recent researches in this field has lead to any-angle path planning algorithms, which are able to find an optimal path in any direction. Theta* [13] has realised an any-angle path

Algorithm 1: Theta* algorithm

Input: s_{start} and s_{goal}
Output: The shortest path from s_{start} to s_{goal} by a set of line segments

```
1  open ← ∅, closed ← ∅, g[s_start] ← 0;
2  parent[s_start] ← s_start;
3  open.insert(s_start, g[s_start] + h[s_start]);
4  while open ≠ ∅ do
5      s ← open.pop();
6      if s = s_goal then
7          return "path found";
8      closed ← closed ∪ {s};
9      for s' ∈ neighbor(s) do
10         if s' ∉ closed then
11             g[s'] ← ∞;
12             parent[s'] ← NULL;
13         g_old ← g[s'];
14         if LOS(parent[s], s') then
15             fp = f(parent[s]) + h(parent[s], s') ;
16             if fp < f[s'] then
17                 parent[s'] ← parent[s];
18                 f[s'] ← fp;
19         else
20             if f[s] + h(s, s') < f[s'] then
21                 parent[s'] ← s;
22                 f[s'] ← f[s] + h(s, s');
23         if g[s'] < g_old then
24             if s' ∈ open then
25                 open.remove(s');
26             open.insert(s', g[s'] + h[s']);
27 return "no path found";
```

algorithm by adding a line-of-sight detection function to each search iteration. Unlike A* or Dijkstra, the path found by Theta* is a connection of line-of sight nodes so that the generated path is smoother, realistic and has fewer turns. With regards to those advantages of Theta*, it has been used as the main global path planning in this paper. The roles of the global path planning with Theta* in the whole system

are described in more details in Section III-A. The pseudo codes of the Theta* is described in Algorithm 1.

C. Dynamic Window Approach

Dynamic window approach (DWA) was introduced by Fox et al. [12] to use multiple constraints of velocity limits, of acceleration limits, and of following the predefined global path into the local path planning. The local searching space is reduced to dynamic windows in a three-step progress. Firstly, DWA considers the robot's trajectories as circular trajectories or curvatures determined by a set of translation and rotation velocities (v_t, v_r). Secondly, only admissible pairs of (v_t, v_r) corresponding to their trajectories are considered if the robot is able to move forward without colliding with obstacles. Finally, the dynamic window limits the admissible velocities to those that the robot can safely reach to the goal in a short time with optimised accelerations. To do so a following objective function given by equation (2) is used to score the admissible velocities. Only the pair of (v_t^*, v_r^*) is selected if the objective function reaches to maximum,

$$(v_t^*, v_r^*) = \underset{(v_t,v_r)}{\operatorname{argmax}} F(v_t, v_r) = \underset{(v_t,v_r)}{\operatorname{argmax}} f\Big(\alpha H(v_t, v_r) + \beta D(v_t, v_r) + \gamma V(v_t, v_r) + \sum_i \omega_i \Omega_i(v_t, v_r)\Big). \quad (2)$$

In this equation, $H(v_t, v_r)$ is a goal heading which is maximised if the robot directly moves towards from the starting point to the goal; $D(v_t, v_r)$ is a distance from the robot to the closest obstacles on the robot's trajectory; $V(v_t, v_r)$ is the forward velocity; and $\Omega_i(v_t, v_r)$ are optional cost functions. The smooth function $f(\cdot)$ is used to smoothen the weighted sum of the above components, α, β, γ and ω_i are the weight of each component.

III. PETRI NET PLANNING AND OBSTACLE AVOIDANCE WITH DIPOLE FIELD AND DWA

A. Overall System

The whole system is built with three main components (Fig. 1). The global path planning utilises the Theta* algorithm to search for the path to the goal for each robotic agent. The paths are formulated as a set of line segments as the typical output of the Theta* algorithm and shared among the team of robots within the working space. By checking the intersections of the planned paths, the PN planning is constructed to control the movements of the robots when they enter intersection regions. The DWA-based local path planning realises moving obstacle avoidance with a dipole field and drives the robots to follow the configured global path to the goal with constraints on the maximum and minimum velocities and accelerations. In the case that one of the robots is getting stuck, the global path planning is reactivated to generate a new path from current position to the goal and PN planning is updated with a new moving control plan.

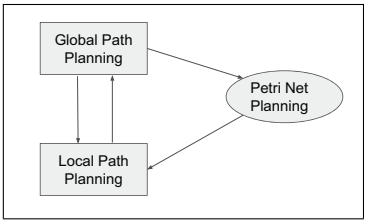


Fig. 1. The overall architecture of the path planning system.

B. Petri Net Planning

The working space is divided into non-overlapping regions called cells where the most simple yet effective way to do this division is to use a grid layout to segment a space with vertical and horizontal cuts. The area of a partition cell is assumed to be small so that not many robots are allowed to enter the cell at any same time to avoid congestion. In this work, only one robot is granted permission to pass through a cell at an instance of time, however in general this solution can be extended to allow more than one robot within a cell. The realisation of the path planning is actually the step-by-step moving of a robot from one cell to another along the predefined trajectory until the robot reaches its goal. Considering a cell as a space resource allocated to a single robot each time, the PN model is constructed by Algorithm 2 to synchronise the movements of multiple robots. In Algorithm 2 (also in Fig. 2), a place of PN is correspondent to a cell and each transition represents a physical connection between two adjacent cells.

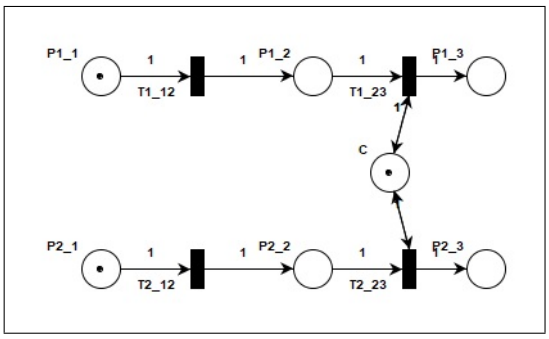


Fig. 2. An example of creating a control place to synchronise the movements of two robots. The places assigned for the moving paths of robot 1 and robot 2 are named by $P1\_\{index\}$ and $P2\_\{index\}$ respectively. The two places $P1\_3$ and $P2\_3$ point to the same area (a cell) thus a control place C is made to manage two robots to pass through the intersection.

The whole system is organised in a centralised manner where a server is used to create PN model for each robot and share information with all robots via communication channels. Once the above PN model is built, the movement of each robot is controlled by firing the enabled transitions and following a sequence of places visited by the token assigned to that robot. Each robot is linked to one token given in Line 6 of Algorithm 2. The movement controlling algorithm for a team of robots is implemented by the Algorithm 3. The PN planning mainly deals with the congestion of robots by preventing several robots to be navigated into a narrow cell at the same time. The prerequisite requirement for this control with PN is that the global planner paths are shared

by the robots. However, this is not applied for humans and other moving obstacles if they do not share their moving trajectories. To avoid those objects, the local path planning with DWA and dipole field is applied.

Algorithm 2: Construction of PN model

Input: A team of robot with initial positions, a set of global paths planned for every robots of the team $\Xi = \{\xi_1, \xi_2, ..., \xi_n\}$, and the working space

Output: The PN model (P, T, W, M_0) of the team

Partition the working space into non-overlapping cells;
Let $P \leftarrow \emptyset$, $T \leftarrow \emptyset$, $W \leftarrow \emptyset$, $M_0 \leftarrow 0$;
for $\xi_i \in \Xi$ **do**
    Find a set of cells $p_i^1, p_i^2, ..., p_i^{n_i}$ which are crossed by ξ_i;
    Add places to PN $P = P \cup \{p_i^1, p_i^2, ..., p_i^{n_i}\}$;
    Add one token into the starting place $M_0[p_i^1] = 1$;
    for $j \leftarrow 1$ *to* $n_i - 1$ **do**
        Add transition $t_{i,j}$ to T;
        $W := W \cup \{(p_j, t_{i,j}), (t_{i,j}, p_{j+1})\}$;
Find all cells $\mathcal{C} = \{c^1, c^2, ..., c^{n_C}\}$ that are passed by at least two paths;
for $c^i \in \mathcal{C}$ **do**
    Add a control place into PN $P = P \cup \{c^i\}$;
    Add one token into this control place $M_0[c^i] = 1$;
    Add transitions to connect c^i with related places with an example given in Fig. 2;

C. Dipole Field for Obstacle Avoidance

The dipole field has been introduced by the authors for obstacle avoidance with moving human or robotic agents [4]. In this method, the magnetic field $\mathbf{B}$ of dipole moment vector $\mathbf{m}$ generated by an agent is given by

$$\mathbf{B}(\mathbf{m}, \mathbf{d}) = \rho(3(\mathbf{m} \cdot \hat{\mathbf{d}})\hat{\mathbf{d}} - \mathbf{m})/d^3 \tag{3}$$

where $\mathbf{d}$ is the distance vector, $d = \|\mathbf{d}\|$, $\hat{\mathbf{d}} = \mathbf{d}/\|\mathbf{d}\|$, in which $||.||$ denotes the norm of the vector, and ρ is a constant. The magnetic moment $\mathbf{m}$ is designed to be aligned with the moving directions of the agent and its magnitude is proportional to the speed of the agent. An agent with the magnetic moment $\mathbf{m_j}$ within the magnetic field $\mathbf{B_k}$ of $\mathbf{m_k}$ would be affected by the force

$$\mathbf{F_{jk}} = \nabla \mathbf{m_j} \cdot \mathbf{B_k} = \rho \nabla \Big(\mathbf{m_j} \cdot \frac{3(\mathbf{m_k} \cdot \hat{\mathbf{d}})\hat{\mathbf{d}} - \mathbf{m_k}}{d^3} \Big) \tag{4}$$

where the gradient ∇ presents the change of potential $\mathbf{m_j} \cdot \mathbf{B_k}$ generated per unit distance, ρ is a constant. To integrate the dipole field into DWA, it is noted that the potential changes reflect the repulsive forces that prevent agents to collide with each other. Therefore, the new term $\Omega_{jk}(v_t, v_r) = -\mathbf{m_j} \cdot \mathbf{B_k}$ is added into Equation (2) to weight the trajectories based on the possibility of collisions. The dipole field term is removed if $\Omega_{jk}(v_t, v_r) < 0$ (not create repulsive forces), and the pair (v_t, v_r) and its trajectory are also removed if they will cause the collisions with another robot or moving objects with regards to the definition of velocity obstacles introduced by Fiorini and Shiller [14].

Algorithm 3: Moving controls

Input: The Petri net model (P, T, W, M_0)

Output: Step-by-step moving strategy for each robot of the team

Let $M \leftarrow M_0$;
Based on the target goals, define the ending marking M_e;
while $M \neq M_e$ **do**
    Find all enabled transitions $\dot{T} \subseteq T$;
    for $t \in \dot{T}$ **do**
        Allow a robot to move if its related token is fired into a new place;
        Use local obstacle avoidance while the robot is on the moving way to the next place;
        Time of arrival of the token to the next place is updated by the moving time of the robot;
        Update the marking M by firing t, $M \leftarrow M + W^+(t) - W^-(t)$;

IV. IMPLEMENTATION AND EXPERIMENTAL RESULTS

A. ROS-based Implementation

The system presented here is implemented in the Robot Operating System (ROS) platform [15] with version Kinetic Kame on Ubuntu 16.04 operating system. As a consequence, the evaluation is performed with the Gazebo simulator (Fig. 3) which provides a realistic simulation for ROS-based robots in both indoor and outdoor environments. The simulated robotic agents are Husqvarna research platform (HRP) [16] mowers with an extra depth sensor, and a LIDAR laser scanner. The robots in the same working space share their locations and global path information through ROS messages. The human actor plugin of Gazebo simulator is customised to allow controlling the moving trajectories of a human subject by either repeated or random patterns. The blob-based detection using the laser scanner developed by the SPENCER project [17] is applied to track the trajectories and velocities of moving human subjects and obstacles in the working space. The PN planning module is programmed in Python with the support of the SNAKES library [18]. Meanwhile, the global path planning with Theta* algorithm and the obstacle avoidance with dipole field on DWA are implemented in C++.

B. Crossing Scenario of Two Robots

The first experiment evaluates the effectiveness of using PN control to avoid collisions of two robots when their planned global paths pass across each other. In Fig. 4.A, the size of the cell in PN planning is configured to 0.2*m*, close to the radius of the circular footprint of the robots, so that

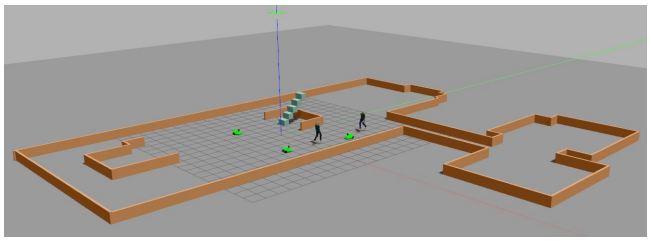
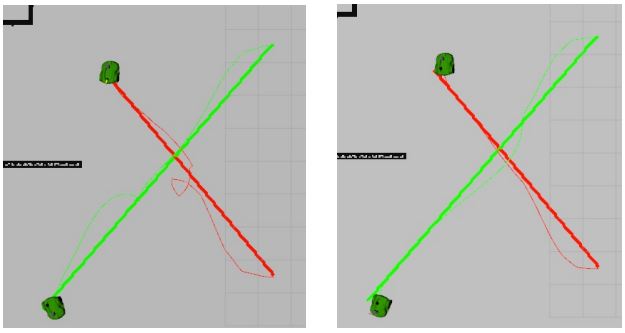


Fig. 3. The Gazebo world with HRP robots and humans in the simulation.

the obstacle avoidance is mainly based on dipole field with DWA. Meanwhile, in Fig. 4.B, that size is configured to 2.5m and the PN control is activated to let one robot stop moving and wait until the second robot pass through the intersection area before moving again. Although no collisions happen,



(A) (B)

Fig. 4. The crossing scenario of two robots, (A) the cell size is small so that PN planning is not applied, (B) obstacle avoidance where PN planning gets involved. The thick solid lines visualise the global paths created by Theta*. The thin lines present the actual trajectories of robots under local path planning. The moving trajectory of Robot 1 is shown in red, while that of Robot 2 is green. The green coloured robots indicate the target locations.

two robots need to travel longer paths in the former case, as shown in Table I, to avoid the collisions with each other. The overall moving time of the two robots to reach their goals is 55s (simulation time from ROS) without PN controls and is reduced to 52s in the other case. A PN to synchronise the movements of two robots created by SNAKES in the later case is visualised in Fig. 5.

TABLE I

TRAVELLING PATH LENGTHS (IN METERS) OF TWO ROBOTS.

	Robot 1	Robot 2
No PN Planning	8.12	10.77
PN Planning	7.45	10.47
Percentage Change	8.25 %	2.79%

C. Crossing Scenario of Three Robots

A similar experiment to the one presented in Section IV-B is performed below with the presences of three robots(6). With PN planning, the travelling paths of the robots are shorter as given by Table II, demonstrating the effectiveness of the proposed approach. Also, with PN the overall

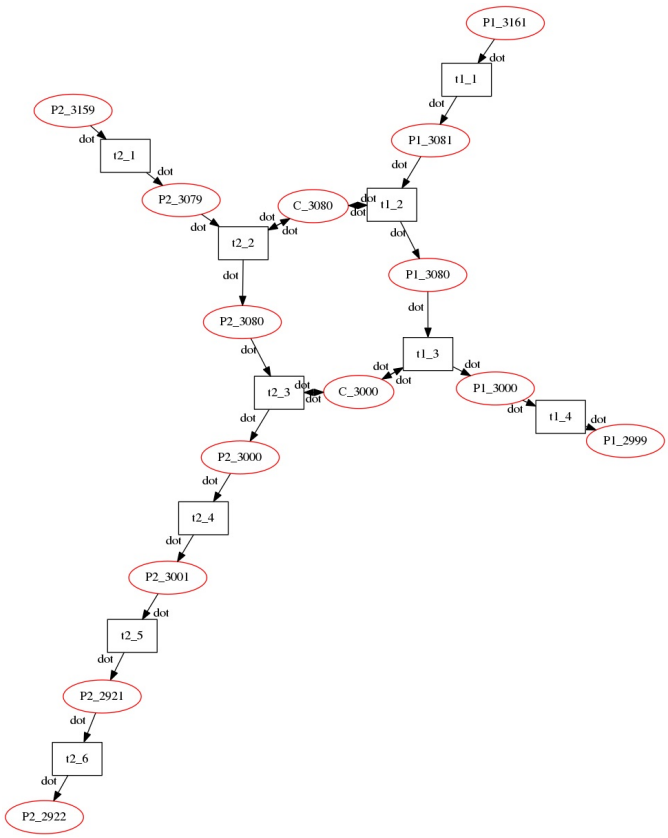


Fig. 5. Generated PN to control the movements of two robots. The places are represented by ovals while the transitions are with rectangles. $P1\_\{cell\_index\}$, $P2\_\{cell\_index\}$, and $C\_\{cell\_index\}$ are of Robot 1, Robot 2, and of control places respectively.

travelling time of all robots is 54s, smaller than that of two robots, 60s, when PN is not used. A part of the PN control (in Fig. 7) consists of the control places to synchronise the movements of two robots and of three robots to pass through an intersection region.

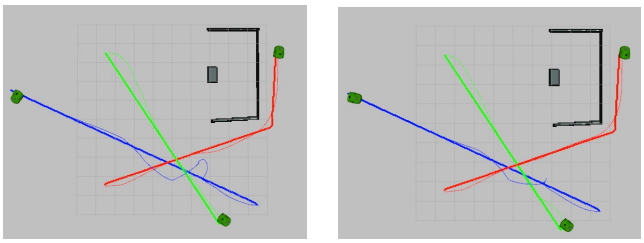
TABLE II

TRAVELLING PATH LENGTHS (IN METERS) AND TIME (IN SECONDS) OF THREE ROBOTS .

	Robot 1	Robot 2	Robot 3
No PN Planning	12.60	10.63	15.61
PN Planning	12.41	10.60	14.49
Percentage Change	1.59 %	0.28%	7.17%

D. Scenario of Robots and Humans Sharing Working Space.

Different to the two previous experiments, in this section, two human actors are added in to the Gazebo world (Fig. 3). The trajectories of three robots are depicted in Fig. 8. Only one robot, green trajectory, stops to wait for the robots to cross the intersection. The other two robots, red and blue trajectories, need to turn to avoid collisions with humans and other robots. Due to the obstacle avoidance with dipole field and DWA, there are no collisions between human and robots nor among robots in the whole experiment.



(A) (B)

Fig. 6. The crossing scenario of three robots, (A) the PN planning is not applied, (B) the PN planning is used. The moving trajectories of Robot 1, Robot 2, and Robot 3 are presented by red, green, and blue respectively. The green colored robots indicate the target locations.

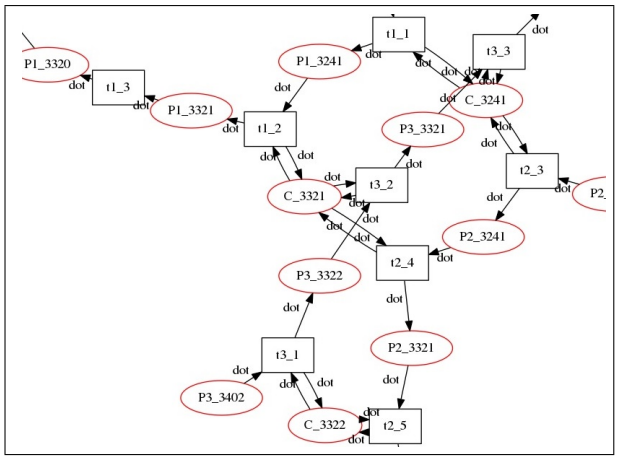


Fig. 7. Generated PN to control the movements of three robots. Only a small part of the full PN is shown here with the same notations as given in Fig. 5.

V. CONCLUSIONS

This paper has introduced a novel path planning algorithm for multiple robots using PN in combination with obstacle avoidance with dipole field and DWA. The experimental results with Gazebo simulator have revealed that the PN control is able to synchronise the movements of multiple robots passing through the intersection, which helps shorten the travelling paths of the robots. Meanwhile, the dipole field implemented on DWA is able to advance the local path planning with an ability to avoid moving humans and other robots in the shared workspace. Using a grid to divide working space as presented in this paper leads to sparse PN with many places to present the moving path of a robot. Meanwhile, only few important places close to the crossing areas of robots should be taken into account. Therefore, future work will focus on the optimisation of the PN presented in this approach.

ACKNOWLEDGMENT

The research leading to the presented results has been undertaken within the research profile DPAC - Dependable Platform for Autonomous Systems and Control project, funded by the Swedish Knowledge Foundation.

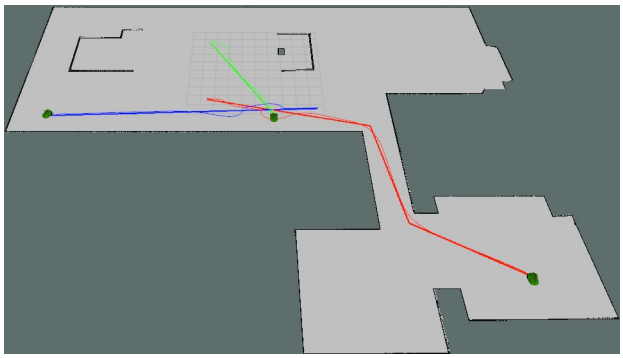


Fig. 8. Moving trajectories of three robots to reach their goals and to avoid collisions with humans and each other.

REFERENCES

[1] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik, vol. 1: pp. 269271, 1959.
[2] P. E. Hart, N. J. Nilsson, and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics SSC4, vol. 4, no. 2, pp. 100107, 1968.
[3] G.-Z. Yang et al., The grand challenges of Science Robotics, Science Robotics, vol. 3, eaar7650, 2018.
[4] T. LanAnh, E. Mikael, and C. Baran, Toward shared working space of human and robotic agents through dipole flow field for dependable path planning, Frontiers in Neurorobotics, vol. 12, 2018.
[5] B. Samanta, and B. Sarkar, Application of Petri nets for systems modeling and analysis, OPSEARCH, 2012.
[6] G. Yasuda, Discrete event behavior-based distributed architecture design for autonomous intelligent control of mobile robots with embedded Petri nets, Advances in Chaos Theory and Intelligent Control, Springer, vol. 37, 2016.
[7] L. Iocchi, M. T. Lazaro, L. Jeanpierre, A-I. Mouaddib, and H. Sahli, COACHES-Cooperative autonomous robots in complex and human populated environments, LNCS Springer, 2015.
[8] P. E. Miyagi, and L. A. M, Riascos, Modeling and analysis of fault-tolerant systems for machining operations based on Petri nets, Journal of Control Engineering Practice, vol. 14, 2006.
[9] B. Lussier, A. Lampe, R. Chatila, F. Ingrand, M. O. Killijian, and D. Powell, Fault tolerant planning: towards dependable autonomous robots, Research Report, LAAS-CNRS, 2015.
[10] J-C. Fabre, M. Lauer, M. Rot, M. Amy, W. Excoffon, and M. Stoicescu, Towards resilient computing on ROS for embedded applications, 8th European Congress on Embedded Real Time Software and Systems (ERTS), 2016.
[11] R. Lill, and F. Saglietti, Model-based testing of cooperating robotic systems using coloured Petri nets, ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems, 2013.
[12] D. Fox, W. Burgard, and S. Thrun, The dynamic window approach to collision avoidance, IEEE Robotics and Automation Magazine, pp. 23-33, 1997.
[13] A. Nash, K. Danial, S. Koening, and A. Felner, Theta*: Any angle path planning on grids, Journal of Intelligent Robot System, vol. 39, pp. 533-579, 2010.
[14] P. Fiorini and Z. Shiller, Motion planning in dynamic environments using velocity obstacles, The International Journal of Robotics Research, vol. 17, no. 7, pp. 760772, 1998.
[15] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, ROS: An open-source robot operating system. In Proceedings of the open-source software workshop, 2009.
[16] https://github.com/HusqvarnaResearch/hrp.
[17] T. Linder, S. Breuers, B. Leibe, and K. O. Arras, On multi-modal people tracking from mobile platform in very crowded and dynamic environments. In Proceedings of the IEEE international conference on robotics and automation, pp. 5512-5519, 2016.
[18] Franck Pommereau, SNAKES: a flexible high-level Petri nets library, Proceedings of PETRI NETS'15, LNCS 9115, Springer 2015.