BUG Algorithm Analysis using Petri net

Alpaslan YUFKA and Aydın AYBAR

Department of Electrical and Electronics Engineering,
Anadolu University, 26470 Eskişehir, Turkey
ayufka@gmail.com and aaybar@anadolu.edu.tr

Abstract

In this study, BUG 1 navigation algorithm which is a simple motion planner, is modelled by the Petri net (PN) for a mobile robot (MR). We prefer this PN model to obtain the discrete data for this algorithm. PN is used as a modelling formalism to define MR's movements such that the step of BUG 1 is constructed by using PN. Thus, all states are obtained the coverability tree. After all coverable marking vectors, the process of PN for BUG 1 is shown step by step on a real MR using the simulation environment of MobileSim.

1. Introduction

In general, navigation is the problem of finding a collision-free motion for the robot system such a mobile robot (MR) from a named place to another in configuration space that is known, unknown or partially known environment [1, 2]. In practically, MR cannot generate a direct motion path from a start (home) point to a goal (destination) point in configuration space so that path planning techniques for MRs must be used in this situation [2]. Many distinct approaches about the navigation are described such that BUG navigation algorithms have presented for MR where they are used in unknown environments and the goal point is reachable. These algorithms are simple motion-planners, and guarantee the reachability to the goal point if it is reachable [1]. They readily construct their own path when an unknown obstacle on the route is encountered. The main aim of these algorithms is generating a collision-free path as MR is navigating. In [4-6], BUG algorithms (BUG 1, BUG 2, Class 1, Alg 1, Alg 2 and DistBug) are studied and compared with each other. In this work, due to the simplicity of BUG 1 navigation algorithm introduced in [3], it is chosen as a case study in order to analyze and model it readily.

Petri net models have been used to represent discrete-event systems (DES) such that DES may be described by the occurrence of events and PN can easily define the dependence of events on other events [12, 14] . In [7], AND/OR logic of PN is used to generate a unique control in order to design cooperative control for multiple robots. Using PN representation, lower-level control of single robot motion and upper-level control of multi-robot cooperation are used for multi-mobile robots based on an event condition system [8]. In [9], PN is used as unified models to represent maps, operation of tasks, and the state of MR. In [10], a Robotic Task Model based on PN is introduced. PN may be a common model to analyze MR's states. In the present work, the representation of the continuous behavior of MR is turned into a discrete form. PN is used as a modelling formalism to illustrate MR's movements. One scenario is considered to show presentation by using PN. It is possible the construct the other scenarios. Thus, an experimental approach is introduced in this work.

This paper presents a Petri net model of BUG 1 navigation algorithm for MRs. BUG 1 algorithm is modelled and examined according to PN's behavioral properties. In order to analyze the BUG 1 by using PN, this model gives a general form the considered algorithm such that any marking vector corresponds a step of BUG 1. Although the motion of MR is a continuous behavior, the representation of this motion is used as a discrete form. Thus, some motions are not considered such as movement during one point to a specific point, getting sensor data continuously, rotation at specific points (start/hit/leave point), etc.. BUG 1 is shown step by step on a Pioneer-P3DX MR on the simulation environment by MobileSim [11].

2. Preliminaries

2.1. Assumptions

In order to implement BUG 1 by PN, assumptions used in the model are:

- In [3], MR's senses are based on tactile sensors but in this study ultrasonic sensors are used as the sensing element.
- In [3], MR is a point object named as the mobile automaton (MA) but in this work MR is a real robot called as Pioneer-P3DX [11].
- Configuration space and objects aren't known by MR.
- The goal point is reachable.
- There are only unknown objects on the route from the start point to the goal point. There is no close boundary of object on the goal point.
- MR never encounters a localization problem and it uses its odometer to localize itself.

In addition, BUG algorithms have three main assumptions: **i)** MR is a point object, **ii)** MR has perfect localization ability, and **iii)** MR's sensors are precise [4-6].

2.2. BUG 1 Navigation Algorithm

BUG 1 procedure is defined in [3]. Based on this; BUG 1 can be applied at any point on a continuous path. The procedure includes registers named as $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, and initially $\mathbf{L_0}$ is assumed as the start (home) point, **S**, and **j** is equal to 1 (**j=1**).

Step-1) A straight line is constructed from $\mathbf{L_{j-1}}$ to the goal point, **G**. MR maintains its motion until one of the following conditions occurs: **i)** MR reaches **G**, and BUG 1 procedure is terminated, **ii)** If an obstacle is encountered, the hit point, $\mathbf{H_j}$, is assigned to the encountered point. Then, jump to step 2.

Step-2) MR starts to circumnavigate the boundary of the obstacle (wall following) by using default local direction (clockwise). BUG 1 procedure is terminated if MR reaches **G** during the boundary following. The register $\mathbf{R_1}$ keeps the shortest distance (Euclidean Distance) from MR's current point at the shortest distance, $\mathbf{Q_m}$, to **G**. The register $\mathbf{R_2}$ keeps the total travelled path related to the obstacle's boundary from $\mathbf{H_j}$. The register $\mathbf{R_3}$ keeps the total travelled path from $\mathbf{Q_m}$. $\mathbf{H_j}$ is reached when MR circumnavigates the whole boundary of the obstacle. A leave point, $\mathbf{L_j}$, is defined as and equal to $\mathbf{Q_m}$. Then, jump to step 3.

Step-3) Using the content of $\mathbf{R_2}$ and $\mathbf{R_3}$, a shortest path to $\mathbf{L_j}$ is determined in pursuance of following the obstacle's boundary. MR determines its local direction (clockwise or counter clockwise) in accordance with this path, and it reaches $\mathbf{L_j}$. New **j** is increased by 1 (**j = j+1**). Then, jump to step 1.

It is noticed that **j** is increased by 1 at the end of each step-3 of the BUG 1 procedure. Hence, **j** is always increased as MR meets a new unknown object on its motion path. **j** is stopped when MR reaches **G**. Based on these, **j** is equal to **K** at the final point where **K-1** is equal to the number of leave points ($\mathbf{L_j}$).

2.3. Petri Net Definition

The Petri net (PN) [12] is a tuple $\hat{G}(P,T,N,O,m_0)$, where P is the set of places ($\forall p \in P$), T is the set of transitions ($\forall t \in T$), N: P x T $\rightarrow \aleph$ is the input matrix that species the weights of arcs directed from places to transitions, O: P x T $\rightarrow \aleph$ is the output matrix that species the weights of arcs directed from transitions to places, and $m_0 : P \rightarrow \aleph$ is the initial marking. Here, $\aleph$ is the set of nonnegative integer numbers.

$M : P \rightarrow \aleph$ is a marking vector, also called as state, M(p) indicates the number of tokens assigned by marking M to the place p. A transition $t \in T$ is enabled if and only if $M(p) \geq N(p, t)$ for all $p \in P$. An enabled transition t may fire at M, yielding the new marking vector:

$$M'(p) = M(p) + O(p, t) - N(p, t) , \forall t \in T, \forall p \in P. \qquad (1)$$

Note that, since t may fire only when $M(p) - N(p, t) \geq 0$, $\forall p \in P$, (1) yields $M'(p) \in \aleph$, $\forall p \in P$. A firing sequence g is a sequence of enabled transitions $\mathbf{t_1 t_2 \ldots t_k} \in T$ where $\mathbf{t_1}, \mathbf{t_2}, \ldots, \mathbf{t_k} \in T$. A marking M' is said to be reachable from M if there exist a firing sequence starting from M and yielding M'. The set denoted by $R(\hat{G}, M)$ is the set of all marking vectors reachable from M. $R(\hat{G}, m_0)$ is the set of reachability. Each element of this set is also an element of the coverbility tree.

3. Petri Net Model of BUG 1 Navigation Algorithm

3.1. Petri Net Model (PN) of BUG 1

Based on $\hat{G}(P,T,N,O,m_0)$, using places, transitions, and relations between them, PN of BUG 1 for MR is shown in Fig. 1. Input matrix is given in (2), the output matrix is given in (3), and the initial marking is $m_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$ ($[.]^T$ denotes transpose of [.]). In Fig. 1, the plain model, in which BUG1 is executed once and no new goal point is given to MR after BUG 1 is terminated, is indicated.

Here the set of places is

$$P = \{\mathbf{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}}\}$$

where $\mathbf{p_1}$ indicates that MR is ready to execute BUG1 task at **S**. ($\mathbf{L_{j-1}}$, initially **j=1**), $\mathbf{p_2}$ indicates that MR is in motion and on the straight line to **G**, $\mathbf{p_3}$ indicates that an object is detected on the route by the ultrasonic sensor, $\mathbf{p_4}$ indicates that no object is detected on the route by the ultrasonic sensor, $\mathbf{p_5}$ indicates that the positioning sensor informs that **G** is reached by MR, $\mathbf{p_6}$ indicates that the positioning sensor informs that **G** is not reached by MR, $\mathbf{p_7}$ indicates that $\mathbf{H_j}$ is assigned by MR, $\mathbf{p_8}$ indicates that MR is in the behavior of boundary following (registers $\mathbf{R_1}$, $\mathbf{R_2}$, and $\mathbf{R_3}$ are running.), $\mathbf{p_9}$ denotes those possibilities that **i)** $\mathbf{H_j}$ is reached by MR (the boundary of the obstacle is entirely circumnavigated by MR.), **ii)** a leaving point ($\mathbf{L_j=Q_m}$) is assigned according to $\mathbf{R_1}$, and **iii)** the local direction and the shortest path to move $\mathbf{L_j}$ is determined by using $\mathbf{R_2}$ and $\mathbf{R_3}$, $\mathbf{p_{10}}$ indicates that $\mathbf{L_j}$ is assigned by MR (j is increased by 1, **j=j+1**), $\mathbf{p_{11}}$ indicates that **G** is reached by MR (BUG 1 task is completed successfully), $\mathbf{p_{12}}$ indicates that the state of the sensor controller where the controller does not allow the ultrasonic and positioning sensors detect at the same time (see assumptions).

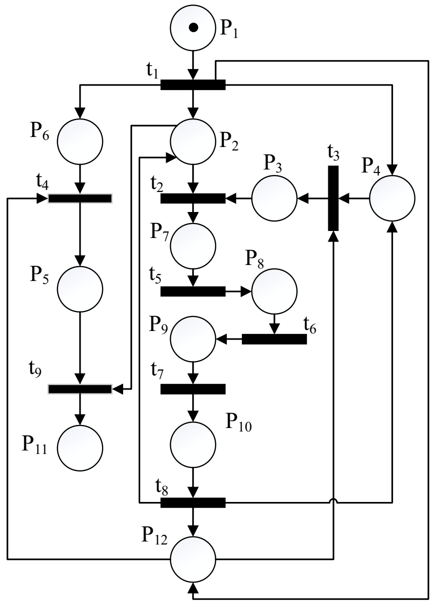


Fig. 1. Plain PN of BUG 1 Algorithm for MR

It is noticed that the ultrasonic sensor senses the obstacles on the motion path in a predefined distance and has two states that are whether the obstacle is detected (the place $\mathbf{p_3}$) or not (the place $\mathbf{p_4}$). In the same manner, the positioning sensor senses the position of MR and has two states that are whether MR is reached **G** (the place $\mathbf{p_5}$) or not (the place $\mathbf{p_6}$). The sensor controller has two states that are whether the ultrasonic sensor or the positioning sensor senses (the place $\mathbf{p_{12}}$ has no token) or not (the place $\mathbf{p_{12}}$ has a token). In addition, **j** is used as a counter which is increased when the loop of the boundary following behavior ($\mathbf{p_2 \rightarrow p_7 \rightarrow p_8 \rightarrow p_9 \rightarrow p_{10} \rightarrow p_2}$) is finished, and it does not affect the boundedness of $\hat{G}$ due to repetition of the same behavior. On the other hand, MR might encounter endless unique objects on the motion path as **j** goes to infinity, but, in the end, MR reaches the goal point if it is reachable, and the mechanism of $\hat{G}$ works.

$$N = \left[\begin{array}{ccccccccc|c}
0&0&0&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&1&0&0\\
0&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&1&0\\
1&0&0&0&0&0&0&1&0&0
\end{array}\right]. \qquad (2)$$

$$O = \left[\begin{array}{ccccccccc|c}
1&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&1&0\\
0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&1\\
0&0&1&1&0&0&0&0&0&0
\end{array}\right]. \qquad (3)$$

The set of transitions is

$$T = \{\mathbf{t_1}, \mathbf{t_2}, \mathbf{t_3}, \mathbf{t_4}, \mathbf{t_5}, \mathbf{t_6}, \mathbf{t_7}, \mathbf{t_8}, \mathbf{t_9}, \boldsymbol{t_{newtask}}\}$$

where $\mathbf{t_1}$ indicates the start of executing BUG1 task by MR, $\mathbf{t_2}$ indicates that MR encounters an unknown obstacle (MR starts circumnavigating the boundary of the obstacle), $\mathbf{t_3}$ indicates that the ultrasonic sensor detects an obstacle on the route, $\mathbf{t_4}$ indicates that the positioning sensor detects MR's location is on **G**, $\mathbf{t_5}$ indicates start of circumnavigating the boundary of the obstacle by MR (wall following), $\mathbf{t_6}$ indicates that MR reaches $\mathbf{H_j}$ which is previously assigned, $\mathbf{t_7}$ indicates start of moving to $\mathbf{L_j}$ by MR, $\mathbf{t_8}$ indicates start of moving from $\mathbf{L_j}$ to **G** by MR, $\mathbf{t_9}$ indicates termination of BUG 1 task by MR. The plain model shown in Fig. 1 includes the set of transitions from $\mathbf{t_1}$ to $\mathbf{t_9}$. If it is desired that a new goal point is given to MR and a new BUG 1 task is repeated all over again after terminating the previous BUG 1 task, $\boldsymbol{t_{newtask}}$ shall be added to the plain PN of BUG1 where $\boldsymbol{t_{newtask}}$ indicates that MR plans a new motion planning of BUG 1 according to new **G**. Due to this difference, $\boldsymbol{t_{newtask}}$ is pointed out in red and italic in T and it should be used in general mode. The addition to the plain PN indicated in Fig. 1 is illustrated in Fig. 2.

In Fig. 2, it is notice that $\boldsymbol{t_{newtask}}$ shall be used if it is desired that a new **G** is given to MR after the previous BUG 1 task. Hence, ingoing and outgoing arcs from $\boldsymbol{t_{newtask}}$ are shown in dash. $\boldsymbol{t_{newtask}}$ shall not be required if it is desired to perform BUG 1 task only once, so that red arcs in dash and the transition $\boldsymbol{t_{newtask}}$ would not be taken into account. In addition, it is considered that the counter **j** is set again as **j=1** described in the BUG 1 procedure after $\boldsymbol{t_{newtask}}$ is fired.

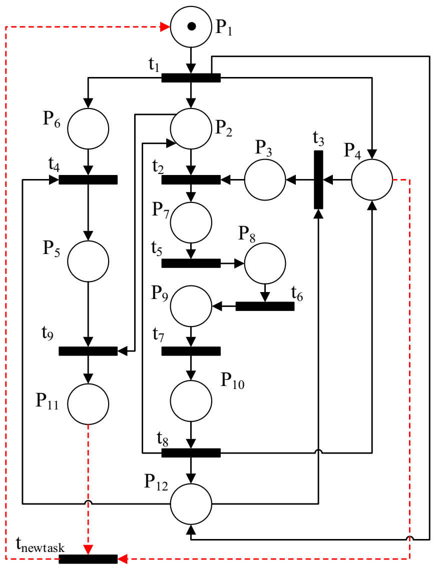
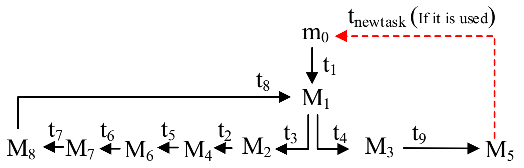


Fig. 2. PN included $\boldsymbol{t_{newtask}}$ of BUG 1 Algorithm for MR

The fundamental behaviors in PN shown in either Fig. 1 or Fig. 2 are: MR generates a collision-free direct motion path from the start point to the goal point if $\mathbf{p_1}$ has a token, MR is on the route of the motion path to the goal point if $\mathbf{p_2}$ has a token, the MR is in the manner of boundary following if the places from $\mathbf{p_7}$ to $\mathbf{p_{10}}$ has a token respectively, and the route is done by MR if both $\mathbf{p_{11}}$ and $\mathbf{p_4}$ has a token.

The set of reachability is $R(\hat{G}, m_0) = \{ m_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8\}$ and the coverability tree for PN of BUG 1 is shown in Fig. 3. Marking vectors are: $M_1 = [0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0]^T$ denotes the execution of BUG1, $M_2 = [0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0]^T$ denotes the detection of an unknown obstacle, $M_3 = [0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0]^T$ denotes the detection of the goal point's location, $M_5 = [0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]^T$ denotes the termination of BUG1, and $M_4 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0]^T$, $M_6 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0]^T$, $M_7 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1]^T$ and $M_8 = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$ denote the behavior of boundary following.



$$\begin{tikzcd}
& & & & & m_0 \arrow[d, "t_1"] & \\
& & & & & M_1 \arrow[dl, "t_3"'] \arrow[d, "t_4"] & \\
M_8 \arrow[urrrrr, "t_8"] & M_7 \arrow[l, "t_7"'] & M_6 \arrow[l, "t_6"'] & M_4 \arrow[l, "t_5"'] & M_2 \arrow[l, "t_2"'] & M_3 \arrow[r, "t_9"] & M_5 \arrow[uul, dashed, "t_{newtask}\ (\text{If it is used})"']
\end{tikzcd}$$

Fig. 3. The Coverability Tree for PN of BUG 1 Algorithm

3.2. Behavioral Properties of PN of BUG 1

In this section, behavioral properties of PN shown in Fig. 1 and Fig. 2 are examined as liveness, deadlock freeness, reversibility, boundedness, and safeness. These are confirmed by using the PIPE (Platform Independent Petri Net Editor) software [13].

The aim and the existence of the transition $\boldsymbol{t_{newtask}}$ are discussed previously. As seen in Fig. 3, If $\boldsymbol{t_{newtask}}$ is used as denoted in PN in Fig. 2, for $\forall M \in R(\hat{G}, m_0)$ and for t ($\forall t \in T$), there exists a firing sequence g such that t can fire at M. PN in

Fig. 1 excluded the transition $t_{newtask}$ would be not live, and the deadlock would be occurred.

A vector K: P → ℵ is K = [1 1 1 1 1 1 1 1 1 1 1 1] where M(p) ≤ K(p), ∀p ∈ P, ∀M ∈ R(Ĝ,m$_0$). PN in either Fig. 1 or Fig. 2 of BUG 1 is K-bounded. λ(K), which indicates the maximal element of K, is 1. Thus, it is also safe.

If $t_{newtask}$ is used as denoted in PN in Fig. 2, m_0 ∈ R(Ĝ, M), ∀M ∈ R(Ĝ, m_0), can be obtained so that PN in Fig. 2 of BUG 1 is reversible. Otherwise, If the transition $t_{newtask}$ did not exist as denoted in the plain PN in Fig. 1, m_0 ∈ R(Ĝ, M) would not be obtained for M_5 ∈ R(Ĝ, m_0) so that PN in Fig. 1 of BUG 1 would not be reversible.

3.3. Application of PN of BUG 1 for MR

BUG 1 navigation algorithm is simulated by PN, and it is shown step by step on a Pioneer-P3DX MR on the simulation environment of MobileSim [11]. BUG 1 application for MR is written in C++ by using ARIA library [11] of Pioneer-P3DX MR in Linux operating system. The test area, which has 10m length, 8m width and two convex-shaped obstacles, is created by Mapper3 [11], as shown in Fig. 4.

In [3], MR is a point object as MA and its senses are based on tactile sensors. In this study, MR is a real MR and it uses ultrasonic sensors as the sensing element. MR never encounters a localization problem and it uses its odometer to localize itself (It has a perfect localization). It has a behavior of wall-following by using range data coming from ultrasonic sensors.

The application carried out in [4, 5] is done again in order to show the operation of PN for BUG 1 algorithm shown in either Fig. 1 or Fig. 2. This operation is shown in Fig. 5.

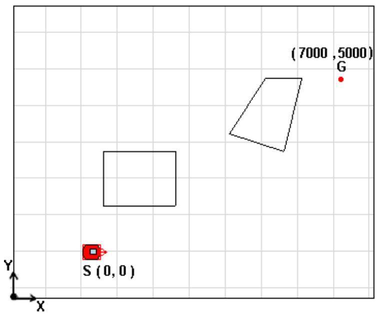
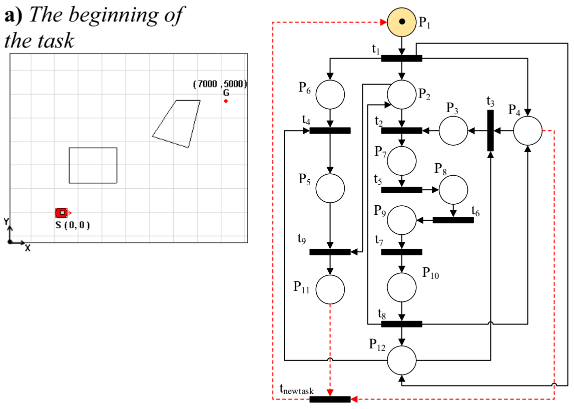
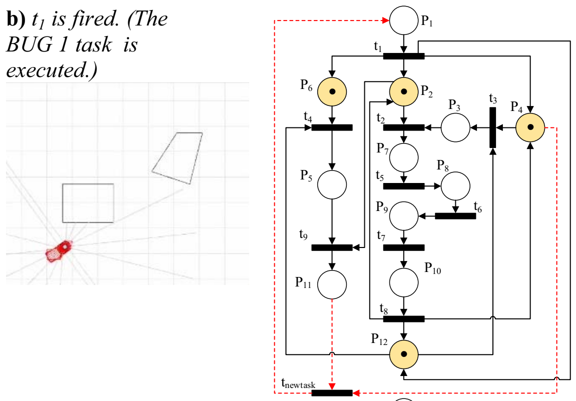


Fig. 4. Simulation Environment by MobileSim

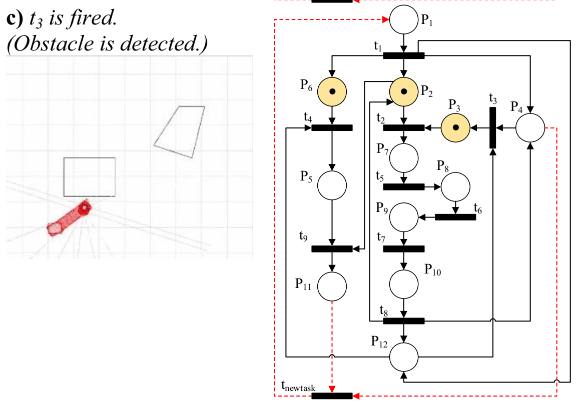
a) *The beginning of the task*



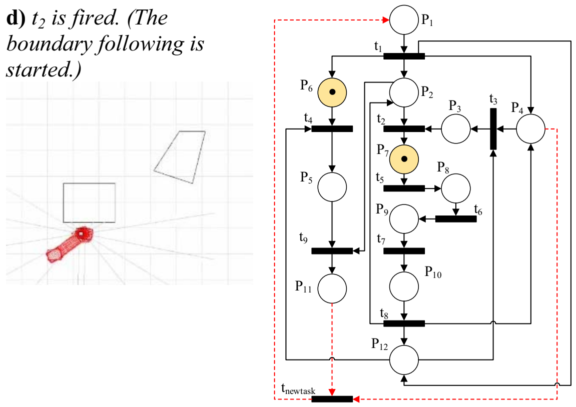
b) *t_1 is fired. (The BUG 1 task is executed.)*



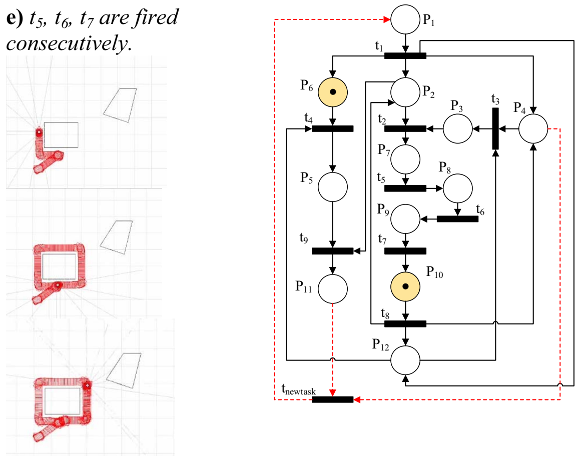
c) *t_3 is fired. (Obstacle is detected.)*



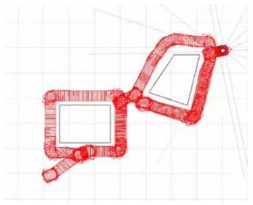
d) *t_2 is fired. (The boundary following is started.)*



e) *t_5, t_6, t_7 are fired consecutively.*



f) *t_4 and t_9 are fired respectively. (BUG 1 task is terminated.)*



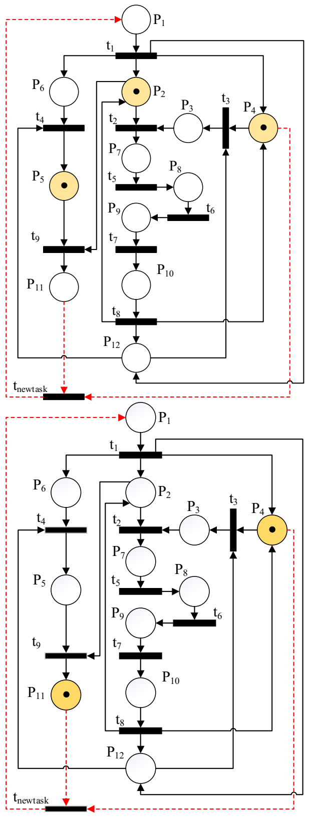


Fig. 5. BUG 1 Algorithm Simulated by PN for the Pioneer-P3DX MR

4. Conclusions

Due to the simplicity of BUG 1 navigation algorithm, a Petri net model (PN) is used to construct BUG 1 for the mobile robot (MR). To implement BUG 1 for MR, the representation of MR's continuous behavior is turned into the discrete form, and PN is used as a modelling formalism to describe MR's movements. Using the simulation environment of MobileSim, BUG 1 is simulated by PN and is shown step by step on MR for a scenerio.

Behavioral properties of PN for BUG 1 are also analyzed. Based on these properties (liveness, deadlock freeness, reversibility, boundedness, and safeness), PN included $t_{newtask}$ where a new BUG 1 task is assigned to MR after firing the transition $t_{newtask}$ at the end of every previous BUG 1's termination is live, deadlock free, bounded, reversible, and safe. So, this provides a connected flow network in PN included $t_{newtask}$. PN excluded $t_{newtask}$ where the BUG 1 task is executed once is bounded, safe, not live and not reversible, and has a deadlock.

The future work may not include modelling the Bug's family (BUG 1, BUG 2, Class 1, Alg 1, Alg 2, DistBug, etc.) in order to demonstrate more than one scenario, but also modelling basic behaviors of MR using PN. In this way, there will be extensive performance. Besides, the structural properties of the Bug's family can be examined. Then, construct a behavioral architecture based on PN, and both lower and upper controllers are designed for this.

5. References

[1] H. Choset, K. M. Lynch. S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, "Principles of Robot Motion: Theory, Algorithms, and Implementations", MIT Press, Cambridge, USA, MA, 2005.

[2] O.Hachour, "Path planning of Autonomous Mobile robot", *International Journal of Systems Applications, Engineering & Development*, vol. *2,* Issue. *4,* pp. *178-190,* 2008.

[3] V.J. Lumelsky and A. Stepanov, "Dynamic path planning for a mobile automaton with limited information on the environment", *IEEE Transactions on Automatic Control*, vol. *AC-31,* No. *11,* pp. *1058-1063,* NOV., 1986.

[4] A. Yufka, and O. Parlaktuna, "Performance Comparison of Bug Algorithms For Mobile Robots", in *5th International Advanced Technologies Symposium (IATS'2009)*, Karabuk, Turkey, 2009, pp. *61-65*.

[5] A. Yufka, and O. Parlaktuna, "Performance Comparison of the BUG's Algorithms for Mobile Robots", in *International Symposium on INnovations in Intelligent Systems and Applications (INISTA'09)*, Trabzon, Turkey, 2009, pp. *416-421*.

[6] J. Ng and T. Bräunl, "Performance Comparison of Bug Navigation Algorithms", *Journal of Intelligent and Robotic Systems*, vol. *50,* issue. *1,* pp. *73-84,* SEP., 2007.

[7] K. N. Lee, and D. Y. Lee, "An Approach to Control Design for Cooperative Multiple Mobile Robots", in *IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC'1998)*, San Diego, USA, CA, 1998, pp. *1-6*.

[8] H. Kashima, and R. Masuda, "Cooperative Control of Mobile Robots Based on Petri Net", in *the 10th International Conference on Advanced Robotics (ICAR'2001)*, Budapest, Hungary, 2001, pp. *339-404*.

[9] K. Kabayashi, and T. Ushio, "An Application of LLP Supervisory Control with Petri Net Models in Mobile Robots", in *IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC'2000)*, Nashville, USA, TN, Hungary, 2000, pp. *3015-3020*.

[10] D. Milutinovic, and P. Lima, "Petri Net Models of Robotic Tasks", in *IEEE Int. Conf. on Robotics and Automation (ICRA'2002)*, Washington, USA, DC, 2002, Vol. *4*, pp. *4059-4064*.

[11] Adept MobileRobots. (2013, OCTOBER 07). MobileSim. Available: http://robots.mobilerobots.com/.

[12] A. Aybar and A. İftar, "Overlapping Decompositions and Expansions of Petri Nets", *IEEE Transactions on Automatic Control*, vol. *47,* No. *3,* pp. *511-515,* MAR., 2002.

[13] Imperial College London. (2013, OCTOBER 07). Platform Independent Petri net Editor (PIPE). Available: http://pipe2.sourceforge.net/.

[14] C. G. Cassandras, and S. Lafortune, "Introduction to Discrete Event Systems", Kluwer Academic, Norwell, USA, MA, 1999.