Error Detection and Recovery Framework for Dependable Navigation

Joong-Tae Park
Dept. of Mechatronics
Korea University
Seoul, Korea
geullu@korea.ac.kr

Tae-Bum Kwon
Dept. of Mech. Eng.
Korea University
Seoul, Korea
haptics@korea.ac.kr

Jae-Bok Song
Dept. of Mech. Eng.
Korea University
Seoul, Korea
jbsong@korea.ac.kr

Woojin Chung
Dept. of Mech. Eng.
Korea University
Seoul, Korea
smartrobot@korea.ac.kr

Abstract − Integration of many basic techniques of navigation such as localization, path planning and motion control is essential for autonomous and dependable navigation. However, if all these navigation components work properly, dependable navigation cannot be guaranteed at all times because it is impossible to predict and deal with many unexpected situations caused by the environmental uncertainties. Therefore, the ability is required for a robot to detect and recover from these unexpected error situations. In this research, we propose a framework using a Petri Net model to detect and recover from several errors occurring during navigation. Various experiments showed that a robot autonomously deals with the unexpected errors and can navigate more reliably in the actual environments.

Keywords − Error recovery framework, Petri Net.

1. Introduction

Various robot technologies such as navigation, human-robot interaction (HRI), manipulation, and so on are needed for a robot to provide good service to humans**.** Although lots of navigation algorithms have been developed, no perfect solutions exist. For service robots to work in a living space, dependable navigation should be ensured for mobile robot navigation.

Dependable navigation is the technology for a robot to move to the goal successfully by itself. To accomplish this task, various navigation components such as localization, path planning, and motion control have to be integrated to work properly. However, although all these basic components work successfully, it is still possible that a robot fails to reach the goal. For instance, if the goal is occupied by a dynamic obstacle for a long time, a robot would just move around the goal or collide with an obstacle. There is a limit to the ability to predict and deal with all kinds of these occasions. It is necessary, therefore, to develop the method that enables a robot to detect the unexpected situations and recover from them. Many problems associated with dependable navigation still remain unsolved although dependable navigation is the most important technology in implementing service robots in the real world.

Minerva was placed in the Smithsonian Institution in the US as a museum tour-guide robot. Minerva worked well even in complex environments [1]. RoboX was a guide robot shown in the national exhibition of Switzerland and operated for 5 months. Various functions were included in this system such as face recognition, range sensor based human detection, and so on [2]. Xavier was a robot that could receive simple orders for navigation through the internet. Xavier also had a function that transmits the images collected during navigation into the web in real time [3].

The above systems focused on autonomous navigation and operated in a structured environment for a limited period of time. However, dependable operation requires a software framework and the sufficient components to cope with unexpected and real situations. Most of service robots have a limited ability to deal with the error situations, so they cannot be employed in a variety of applications.

Various error situations occurring in real environments can be modeled using discrete events and the state of the system changes by these discrete events. Thus, the state of the robot is identical to that of the discrete event system that is changing by the discrete event at each time. In this paper, we propose the error detection and recovery framework that adopts a Petri Net model among many modeling methodologies for discrete event systems.

The remainder of this paper is organized as follows. Section 2 briefly presents Petri Net and a framework for error detection and recovery. Section 3 deals with the proposed model using the analysis method based on Petri Net and discusses performance of the proposed framework by experiment results. Section 4 presents conclusions.

2. Petri Net Model of Error Detection and Recovery Framework

2.1 Petri Net

A Petri Net (PN) visually models dynamic conditions of a discrete event system, so that it is easier to understand the present conditions of the system. It can also represent parallel events and synchronized events. Moreover, it can systematize and unify the control scheme of a system. Due to these characteristics, PN has been studied in the field of control or process planning which requires the flexibility of the system [4][5]. Fig. 1 shows the basic elements for modeling.

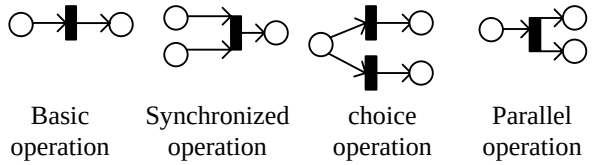

Fig. 1. Modeling primitives used in Petri Net.

PN generally represents an event by a transition. To generate a transition, several conditions of the system should be satisfied, and the information on each condition is stored in a place. A place can indicate either the input for the transition to occur or the output of the transition. PN uses the concept of a token to describe the behavior of the system. A token means the condition for each place and "marking" is to allot a token or tokens to the places. As described above, the place contains information on the event occurrence condition and the condition of the system is described by the number of tokens in that place. The structure of PN is defined as follows:

Definition 1. PN is defined as a 5-tuple as follows:

$PN = (P, T, A, w, m_0)$

where

P is a finite set of places,

T is a finite set of transitions,

A is a set of arcs, a subset of the set $(P \times T) \cup (T \times P)$,

w is a weight function, w: A → {1, 2, 3…},

m_0 is an initial marking.

2.2 Error Detection and Recovery Framework

In this research, several unexpected situations and their solutions are modeled using PN, and then the error detection and recovery framework is implemented. This framework can be applied to the navigation system like Fig. 2. The architecture of a navigation system shown in Fig. 2. was modified from the architecture applied to other indoor service robots. The architecture consists of a MCL based localization module, a gradient method based path planning module, a DWA (Dynamic Window Approach) based motion control module, and so on [6][7][8].

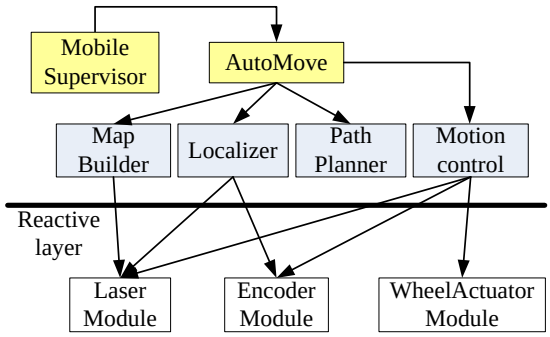

Fig. 2. Architecture of navigation system.

When a robot receives the order to move to the goal, the Mobile Supervisor module activates the navigation system. In Fig. 3, the location of the token changes from P0 to P1. As the token moves to P1, the robot requests the localization module to send the information on its pose. The localization module fires event t2 if the kidnapping occurs, but otherwise event t1. When event t1 is reported from the localization module, the robot gets information on its pose by local localization. After that, event t4 enables the AutoMove module to work.

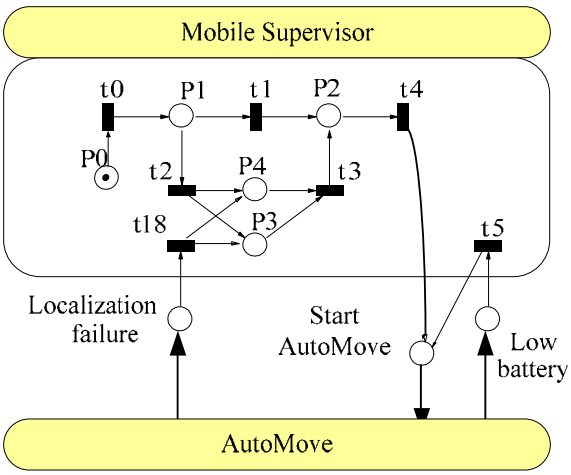

Fig. 3. Net model of mobile supervisor.

Table 1 Description of Fig. 3.

P/t	Description
P0	System idle.
P1	Initial localization.
P2	Estimated robot pose.
P3	Start wandering.
P4	Start global localization.
t0	System on.
t1	Estimated pose by local localization.
t2	Kidnapping
t3	Estimated pose by global localization.
t4	Start navigation.
t5	Go to charging station.
t18	Start global localization.

If the Mobile Supervisor module starts navigation, a token is placed at Start AutoMove (P6), as shown in Fig. 4. The AutoMove module then fires the navigation start event (t6) and the token moves to P7 indicating that the navigation is in normal operation. After that, the robot travels to the goal. If the robot encounters the error situations (t7, t8, t11, t13) that can occur during navigation, the AutoMove module starts to recover from that situation through the designed framework. For instance, when the goal is occupied by a dynamic obstacle (t8), the token moves from P7 to P9. The AutoMove module generates a temporary goal by the designed framework, and then the robot gets there. If the dynamic obstacle disappears and

the event t9 occurs while the robot navigates, it moves back to the original goal. Thus, the state of the AutoMove module changes from P9 to P7 and it indicates that the navigation is in normal operation. However, if the dynamic obstacle occupies the original goal continuously after the robot reaches the temporary goal, the robot stays there for a while. When the situation is not changed as time goes by, the abnormal termination event (t10) occurs. The state of the AutoMove module is changed from P9 to P10 and navigation is completed after reporting the abnormal termination event to the Mobile Supervisor module.

The advantages of the proposed framework can be summarized as follows:

1) It deals with the error situations effectively that are reported from several navigation modules.

2) It detects and recovers from the error situations autonomously that can occur during navigation.

3) It is designed to solve the current problems and also to be easily modified to deal with possible problems or additional demands.

4) It enables the state of the robot to be easily understood and analyzed by visualization using PN.

5) It prevents design errors that can cause malfunctions because it can be analyzed by the mathematical method.

Table 2 Description of Fig. 4.

P/t	Description
P6	Start navigation.
P7	Normal navigation.
P8	Low battery.
P9	Go to temporary goal.
P10	Abnormal termination of navigation.
P11	Path re-planning.
P12	Path re-planning failure: no path to goal.
P13	Navigation completed.
P14	Localization failure.
t6	Start navigation.
t7	Low battery.
t8	Goal occupied by moving obstacle.
t9	Moving obstacle removed from original goal.
t10	AutoMove terminated due to goal occupied by obstacle.
t11	Obstacle on path.
t12	Path planner finds a path to the goal.
t13	Path planner cannot find a path to the goal.
t14	Reach the goal.
t15	Fault: Process completed.
t16	Process completed.
t17	Localization failure.

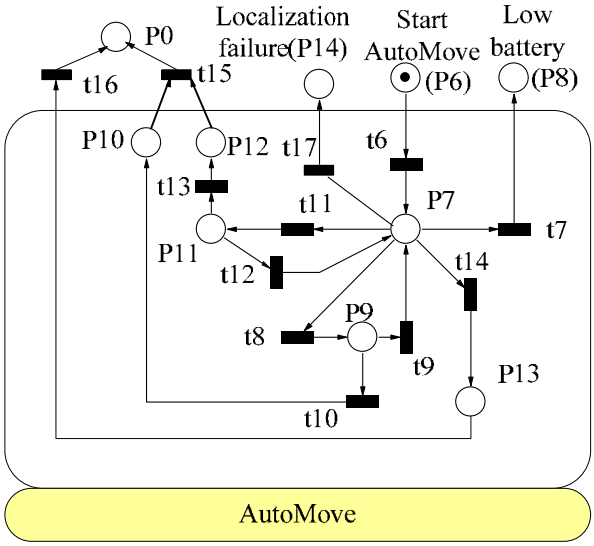

Fig. 4. Petri Net model of AutoMove.

3. Analysis and Experiments

3.1 Analysis of PN Model

This section shows that the proposed model satisfies the basic properties of PN by using the reachability method. The fundamental properties of PN are as follows:

1) If the number of tokens does not exceed one and no output places of transitions are empty, then the safeness property is satisfied.

2) If the number of tokens does not exceed k, then the boundedness property is satisfied.

3) If a certain number of tokens always move over all states, then the conservation property is satisfied.

4) If it is possible to fire transitions continuously according to the firing sequence from the initial marking M_0, then the liveness property is satisfied.

5) If there is any firing sequence which changes the system state from marking M_0 to marking M_n, then the reachability property is satisfied.

6) If marking M_0 can be reached from every marking M that can be reached from M_0, then the reversibility property is satisfied.

The reachability graph shown in Fig. 5 describes the distribution of tokens that change continuously by the transitions. As the number of tokens in each place of PN shown in Fig. 6 does not exceed one when the initial number of token is 1, it satisfies the boundedness property and the safeness property. A ceratin number of tokens move after n cycles in the PN model and thus it satisfies the conservation property. Since the movement of tokens is not blocked due to deadlock, as shown in Fig. 5, it satisfies the liveness property. As it is possible to reach any place, the reachability property is also satisfied. It maintains the same initial value after n cycles, so it

satisfies the reversibility property. Therefore, the proposed model satisfies all properties of PN.

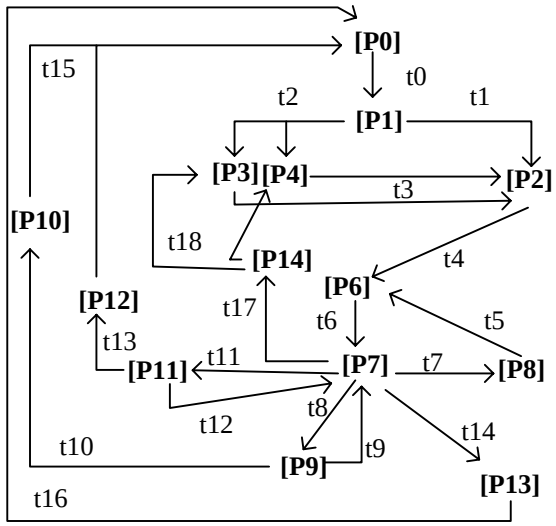

Fig. 5. Reachability graph.

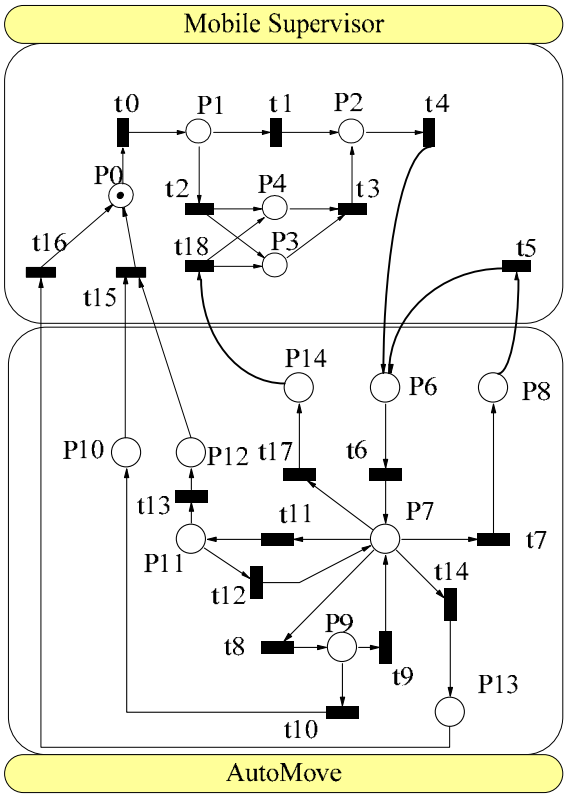

Fig. 6. Petri net model of framework.

3.2 Experiments of Error Detection and Recovery

This experiment shows the error detection and recovery process when the battery voltage is low. As shown in Fig. 7, the robot moves toward a goal and the battery voltage drops below 20V during navigation. Then the robot stops moving to the goal and moves to the battery charging station.

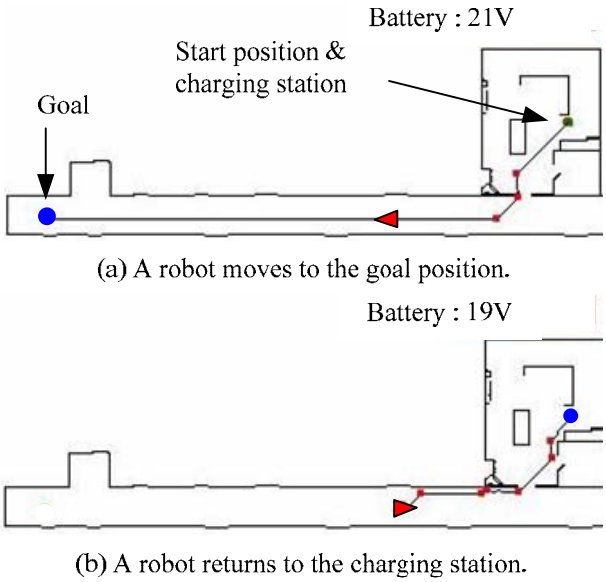

Fig. 7. Experimental results of error recovery from low battery voltage.

The next experiment also shows the process that the robot detects and recovers from the error situation through the proposed framework. While the robot moves toward the goal in Fig. 8(a), a dynamic obstacle occupies the goal. Then the robot sets a temporary goal within 2m from the original goal and stays there until this dynamic obstacle goes away, as shown in Fig. 8(b). After the robot detects the dissapearance of the obstacle, it restarts to move toward the original goal, as shwon in Fig. 8(c).

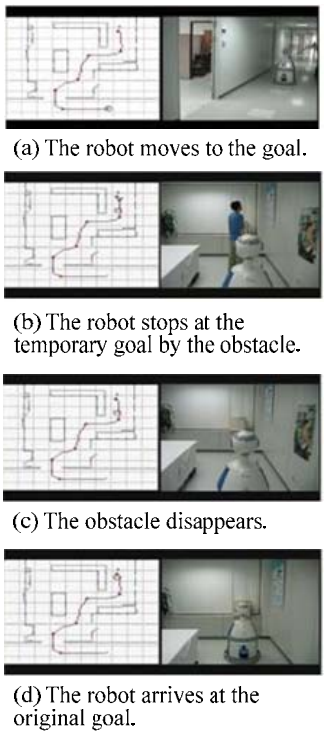

Fig. 8. Experiments showing recovery from goal occupation caused by moving obstacle.

4. Conclusions

In this paper, a framework was proposed to detect and recover from the error situations that occur discretely. The proposed framework was modeled using PN. The performance of the framework was validated by a series of experiments. From this research, the following conclusions have been drawn.

1) Dependability of mobile robot navigation is improved through the proposed framework that can detect and recover from error situations during navigation.

2) The proposed framework can be used practically because it models the error situations and their recovery methods which occur commonly in various environments.

3) The PN model can effectively deal with the discrete events occurring during mobile robot navigation.

Acknowledgements

This research was performed for the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy of Korea.

References

[1] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, and D. Fox, “MINERVA: A Second-Generation Museum Tour-Guide Robot,” Proc. of the IEEE Conference on Robotics and Automation, pp. 1999-2005, 1999.

[2] R. Siegwart, K.O. Arras, S. Bouabadallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, M. Meisser, R. Philippsen, R. Piguet, G. Ramel, G. Terrien, and N. Tomatis, “Robox at Expo.02: A Large-scale Installation of Personal Robots,” Robotics and Autonomous Systems, Vol.42, No.3-4, pp.203-222, 2003.

[3] R. Simmons, R. Goodwin, S. Koenig, J. O’Sulliban, and G. Armstrong, Beyond Webcams: An Introduction to Online Robots, MIT Press, MA Cambridge, 2002.

[4] P. Freedman, “Time, Petri Nets, and Robotics,” IEEE Transactions on Robotics and Automation, Vol. 7, No.4, pp. 417-433, 1991.

[5] F. Norelis, R. Chatila, “Control of mobile robot actions,” Proceeding of IEEE International Conference on Robotics and Automation, Vol.2, pp. 701 – 707, 1989.

[6] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, “Monte Carlo localization: Efficient position estimation for mobile robots,” Proceeding of International Conference on Artificial Intelligence, pp. 343-349, 1999.

[7] K. Konolige, "A Gradient Method for Real-time Robot Control," Proceeding of International Conference on Intelligent Robots and Systems, pp.639-646, 2000.

[8] D. Fox, W. Burgard, and S. Thrun. “The Dynamic Window Approach to Collision Avoidance,” IEEE Robotics and Automation Magazine, pp23-33, 1997.