Experimental Research of Navigation Behavior Selection Using Generalized Stochastic Petri Nets (GSPN) for a Tour-Guide Robot

*Gunhee Kim, †Woojin Chung, *Sung-Kee Park, and *Munsang Kim

*Intelligent Robotics Research Center
Korea Institute of Science and Technology
39-1 Hawolgok-dong, Sungbuk-ku, Seoul, 136-791, Korea
{knir38, skee, munsang}@kist.re.kr

†Department of Mechanical Engineering
Korea University
Anam-dong, Sungbuk-ku, Seoul, 136-713, Korea
smartrobot@korea.ac.kr

Abstract - **This paper proposes a formal selection framework of multiple navigation behaviors for a service robot. In our approach, modeling, analysis, and performance evaluation are carried out based on the Generalized Stochastic Petri Nets (GSPN). By adopting probabilistic approach, our framework helps the robot to select the most desirable navigation behavior in run time according to environmental conditions. Moreover, after a mission, the robot evaluates prior navigation performance from accumulated data, and uses the results for the improvement of future operations. Also, GSPN has several advantages over classic automata or direct use of Markov Process. The basic ideas of the framework were introduced in our previous work [1]. Thus, this paper focuses on experimental verification by implementing the framework into the guide robot *Jinny*. We conduct the experiments about real guidance tasks with visitors in the *National Science Museum of Korea*. The results show that the proposed strategy is useful to select an appropriate navigation behavior in a dynamic space.**

Index Terms – Robot navigation, behavior selection, a guide robot, Generalized Stochastic Petri Nets, Monte-Carlo localization

I. Introduction

A guide robot is one of the most emerging fields of service robot applications. The KIST (Korea Institute of Science and Technology) has developed a guide robot *Jinny* for permanent installation at the *National Science Museum of Korea* as shown in Fig.1 [2]. The *Jinny* was implemented by using a Petri net (PN) based control architecture, which was designed for multi-functional service robots in our previous work [3] [4]. In addition, we developed an integrated navigation system, which was successfully applied to service tasks [5].

Our experiences for permanent installation taught us that it is advantageous that a guide agent uses several navigation behaviors in order to operate robustly in a dynamic and unmodified environment. For example, in general cases, it is advisable that the robot uses a map-based navigation. However, if the robot is in a narrow or crowded region between exhibits and visitors, the sensor-based navigation, like a contour tracking, is much dependable since it is less affected by localization accuracy and uncertainties of environment.

Navigation using multiple behaviors has been widely accepted in biomimetic study [10]. It is obvious not only for humans but also for simple animals like bees and ants. In [10], a desert ant *cataglyphis* is explained as a typical example. The ants rely on the outdoor navigation using the polarization patterns of the sky. However, such a strategy is useless in their nest, thus the ants call for another navigation strategy like using pheromones.

In general, navigation is accomplished by the cooperation of several key components like a localizer and a path planner. Thus, the behavior selection problem is influenced by the information provided by these navigation components. However, as the considered components and navigation behaviors increase, it becomes troublesome to manage the relationships between them. Therefore, it is necessary to have a formal or mathematical framework which handles this issue. Moreover, the behavior selection problem usually has nondeterministic properties. Thus, it should be solved not by pre-defined logics but by performance estimation in run-time.

This paper deals with a formal selection framework of multiple navigation behaviors for a guide robot. The basic ideas of the framework and simulation results were introduced in our previous work [1]. This paper focuses on experimental verification by implementing the developed framework into the *Jinny*. Experimental results were gathered from real guidance tasks with visitors in the *National Science Museum of Korea*. The results clearly show that the proposed strategy is useful to select an appropriate navigation behavior in a dynamic space.

Fig. 1 The guide robot *Jinny* at the *National Science Museum of Korea*.

II. Overview of the Proposed Framework

In our approach, modeling, analysis, and performance evaluation are carried out based on the Generalized Stochastic Petri Nets (GSPN). By using probabilistic approach, our framework selects the most desirable navigation behavior in run time according to environmental conditions. Moreover, the robot evaluates prior navigation performance from recorded data, and uses the results for the improvement in future tasks. This approach is particularly useful for the most service robots which repeatedly operate in the same workspace.

GSPN is useful in our application since the activities of the selection framework are event-driven and can be divided into the possible discrete states. Our strategy has following three major advantages by using the formalism. First, our framework is developed on firm mathematical foundation. This makes it possible to set up state equations and other mathematical models governing the behaviors of a system. Therefore, it allows not only to fully identify qualitative behavioral properties of designed logics, but also to carry out quantitative performance analysis. Second, our method supports modular and incremental designs of navigation framework since GSPN has powerful modeling ability. It can model concurrency, asynchronous events, logical priority relations, and structural interactions. Third, as a graphical tool, GSPN can visualize both static and dynamic aspects of a system. The dynamic changes of states can be explicitly described since the GSPN exploits tokens, which clearly indicate state transitions with respect to event firings.

GSPN are stochastic nets in which the delays are probabilistically specified. The basics of GSPN are briefly reviewed in [1]. The details of Petri net theory can be found in many references like [7], [8], and [9]. It has been shown that GSPN models, including extensions such as priority transitions, random switches, inhibitor arcs, and probabilistic arcs, can be converted into their equivalent Markov process (MP) representations [9]. MP's can be obtained from the reachability graph of the PN's, which is one of the powerful structural analysis methods for PN's. For example, it formally identifies deadlocks, mutual exclusion, liveness, and boundedness. This structural analysis prevents developers' design mistakes which can cause malfunctions of a robot.

Through the analysis of the derived MP's, it is possible to find various performance estimates of a system, which includes the probability of a particular condition, the expected value of the number of tokens, the mean number of firings per time unit, and the mean system throughput [7]. Our framework computes these performance estimates for the selection of navigation behaviors.

From the view of control logic design, there are some benefits for choosing Petri net over classic automata used in other systems [11] or in UML [12]. First, automata are inadequate to describe the concurrence of activities in the system. On the other hand, tokens in PN's can explicitly monitor the statuses of several components simultaneously. It means that automata only represent the status of the entire system at a certain instance, whereas the PN's can describe statuses of components individually. Second, the qualitative or quantitative analysis for systems modeled by PN's is more powerful than ones by automata.

The PN based approach is also advantageous over direct use of MP's like Partially Observable Markov Decision Process (POMDP) models [13]. GSPN are isomorphic to embedded Markov Chains [9]. The main advantage of PN is that the number of places and transitions only increases linearly as the system complexity increases, while the number of states in the MP's increases exponentially. Also, there is no need to manually enumerate all the possible states of MP's since they are automatically generated from the GSPN model. This frees the modeler from having to painstakingly account for all possible states of the system. Thus, modular design and incremental changes to a PN models can be carried out simply by adding tokens, places, or transitions. On the other hand, minor changes in a MP model usually require redefining all the states in the model.

Due to PN's mathematical superiority, the overall processes mentioned above, including the derivation of the reachability graph and the equivalent MP model, the computations of steady-state probabilities, and simulations are automated and incorporated into free or commercialized software tools such as Design/CPN [15] and TimeNet [16].

III. GSPN Based Modeling and Performance Estimation

A. *Problem Statement*

The *Jinny*'s navigation system is a range sensor based scheme without modification of an environment [2]. Range sensors are used for mapping, path planning, and localization. Our localization method is a probabilistic map-matching scheme based on Monte Carlo localization [6].

In our system, the localizer has two internal states, *Success* and *Warning*. the localizer *Warning* event is fired when the localizer fails in estimating robot's accurate position for two consecutive iterations. The failure does not mean that the localizer completely loses its position but that it gives warning about two repetitions of matching failures. On the contrary, the localizer *Success* event is mapped to two iterations of success in estimating the robot's position. It means that the matching between the map and measured data at the pose estimated by the localizer is more accurate than that at the point obtained by odometry.

We implemented four navigation behaviors into the *Jinny* [2]. We learned by experience that this approach is more robust and reliable in uncertain and dynamic surroundings than the method using a single navigation strategy. In this paper, we mainly consider two types of navigation behaviors, *AutoMove* and *Contour tracking*. The detailed description of these motions is summarized in Table I.

The *AutoMove* is our fundamental navigation strategy which has the path planner using modified Konolige's gradient method [17]. The advantages of the *AutoMove* are generality and optimality. It is applicable in any situations, and it also makes it possible for a robot to move the desired position with shortest collision-free trajectory. The path planner of the *AutoMove* is a time-consuming since it computes the navigation function of each grid cell of a whole workspace. This property places a limitation on map size. In our system, the map is set with a 10m × 10m, and it takes about 0.3 second on average to execute one cycle of the path planning algorithm. Due to the restriction on a map size, the moving distance does not exceed 10m by using a single execution of *AutoMove*. Thus, if the robot should go to a far-off position, the intermediate goal set is generated. Then, by iterating *AutoMove*, the robot can reach desired position.

Our path planner copes with two errors, *Path blocking* and *Goal occupation*. Such situations frequently occur in a human co-existing environment. The *Path blocking* implies that there is no path between two adjacent nodes on the 10m × 10m map. The *Goal occupation* means that the goal is temporarily occupied by a dynamic object. Since both exceptions are largely caused by people, if they arise, the robot stops moving and say "Step aside, please." Consequently, the path planner also has two internal states, *Normal* and *Abnormal*, similarly to the localizer. If these two kinds of errors occur, the path planner is transited to the *Abnormal* state.

On the other hand, the *Contour tracking* is a wall following technique makes the robot move smoothly with a fixed distance to the wall. It generates velocity commands based on only raw laser sensing data in every sampling time. Thus, the *Contour tracking* has a better localization property than the *AutoMove*. The performance of the *Contour tracking* is less affected by localization accuracy since it is a sensor-based navigation. It works well even if the localizer fails. Also, the *Contour tracking* improves localization reliability, as known as a coastal navigation paradigm [14]. As the robot navigates along the wall, it is more likely that the robot collects accurate environmental information. Thus, it can decrease the likelihood of getting lost.

TABLE I DESCRIPTION OF TWO NAVIGATION BEHAVIORS

Type	*AutoMove*	*Contour Tracking*
Algorithm	- Shortest path planning with obstacle avoidance	- a (left, right, center) wall-following technique using only laser scan data
Merits	- Optimality (shortest path to any points on the maps) - Generality (applicable in any situations)	- Reactive - Rise localization reliability - Less affected by localization accuracy
Desirable environment	- Generally applicable, but the performance drops in a narrow or crowded region.	- An area where there are many static feature like walls or exhibits

From this observation, we make one rule for the behavior selection; if the localizer falls into the *Warning* state, *Contour tracking* is unconditionally selected.

The main problem tackled in this paper is to select one navigation behavior out of two behaviors according to environmental conditions. The criterion of this selection problem is "which behavior leads the robot to a goal faster than the other with guaranteeing localization safety." Obviously, the *AutoMove* is more efficient than the *Contour tracking* since the moving distance is shorter in most cases. However, if the area is so crowded that the localization *Warning* takes place too frequently, the *AutoMove* may not be advantageous. Using *Contour tracking* from the beginning to end may be more efficient than using *AutoMove*. Also, the *Contour tracking* can reduce the probability of losing the position. Moreover, the errors of the path planner can lower the performance of *AutoMove*. Therefore, the behavior should be selected by considering both the traveling distance and estimated success rates of the localizer and the path planner.

B. *Modeling Method*

The modeling method goes through following procedure. First, based on a given system description, navigation behaviors and required components are identified. Behaviors are designed as places, and the switches between them are modeled as transitions. Each component is modeled as an independent GSPN model. Behavior models and component models are used as basic building blocks for the entire model. And then, they are linked by transition and arcs, according to the relationships between them. The behavior model and each component model have its own token in order to clearly express its internal status independently.

Fig.2 shows the resultant GSPN model for the behavior selection framework. Table II describes the physical meaning of places and transitions of the model. The GSPN model has eight places, twelve timed transitions (drawn as white bars), and three immediate transitions (as black bars). The initial marking is M_0=(1,0,0,0,0,1,0,1,0), which are denoted as $P_0P_5P_7$ in reachability graph (see Fig.3.(a)) by specifying the places having a token. Tokens assigned to P_5 and P_7 indicates that it is assumed that the localizer initially knows its position and the path planner initially is in a normal state.

Two navigation behaviors, *AutoMove* and *Contour tracking*, are modeled as P_1 and P_2 respectively. Initially, the robot selects its motion by a random switch comprising the transitions t_0 and t_1 with corresponding probabilities p and 1-p. The transition between them takes place according to the change of localizer states. The immediate transition t_3 means that the robot takes *Contour tracking* as soon as the localizer *Warning* event fires. The other transition between two behaviors, t_2 and t_4, are modeled as timed transitions in order to express that the robot can change its current behavior during the localizer *Success* state, if necessary.

The place P_4 models the failure of a navigation task. It is due to following two cases. One is that the path planner cannot detect a path to the goal for a long time during *AutoMove* (designed by t_{11}), and the other is that the localizer completely lose its position during the *Contour tracking* (depicted by t_{12}).

One of the most important modeling issues is how to set the firing rates $\Lambda=\{\lambda_0,\cdots,\lambda_9\}$. The physical meanings of these parameters will be illustrated in Chapter IV. In order to perform analytic evaluation of GSPN designs, it is necessary to obtain an embedded Markov chain (EMC). Fig.3.(b) shows the EMC induced from the rechability graph of Fig.3.(a), which is derived from GSPN model of Fig.2.

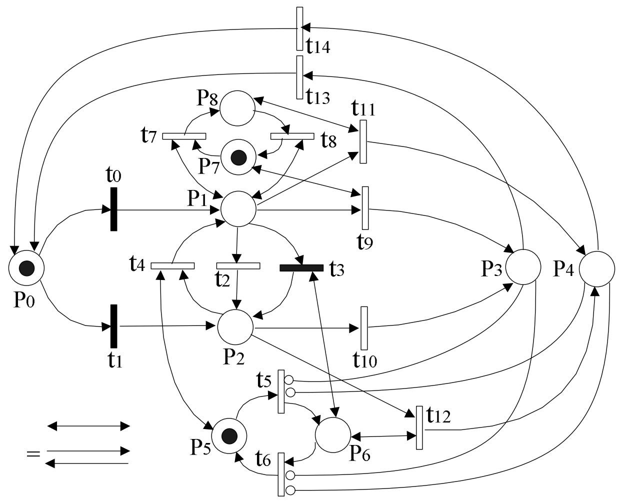
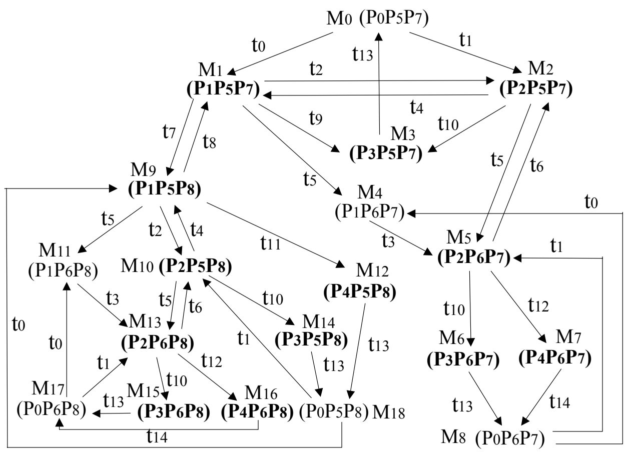


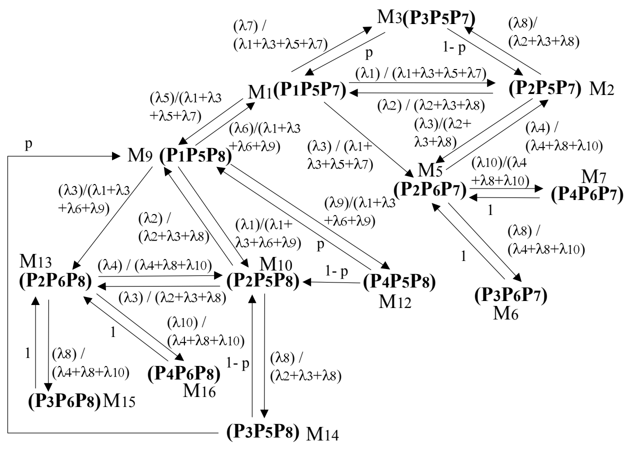
Fig. 2 A GSPN model for performance estimation.

TABLE II DESCRIPTION OF PLACES AND TRANSITIONS

Place	Description	
P0	Navigation available	
P1 (P2)	Running *AutoMove* (*Contour tracking*)	
P3 (P4)	Completion *Success* (*Failure*)	
P5 (P6)	Localization *Success* (*Warning*)	
P7 (P8)	State: Path planner *Normal* (*Abnormal*)	
Transition	Description	Firing rate
t0	Start *AutoMove* (prob. *p*)	-
t1	Start *Contour tracking* (prob. 1-*p*)	-
t2 (t4)	Convert to *Contour tracking* (*AutoMove*) due to performance estimation	λ_1 (λ_2)
t3	Convert to *Contour tracking* due to localization *Warning*	-
t5 (t6)	Localization *Warning* (*Success*) event fired	λ_3 (λ_4)
t7 (t8)	Path planner *Normal* (*Abnormal*) event fired	λ_5 (λ_6)
t9 (t10)	*AutoMove* (*Contour tracking*) completed	λ_7 (λ_8)
t11 (t12)	Task Failed due to no path to the goal (failure of *contour tracking)*	λ_9 (λ_{10})
t13, t14	Initialization	λ_{11}



(a) Reachability graph



(b) Reduced embedded Markov chain

Fig. 3 Derived models for performance estimation.

Fig.2 and Fig.3 clearly shows the advantages of PN over direct use of automata or MP. The PN model (Fig.2) are more intuitive than the automata model (Fig.3(a)), since PN model independently describes the statuses of behaviors and the localizer, whereas the automata model represents only the status of the entire system. Also, GSPN does not go through exponential state explosion and the MP model (Fig.3(b)) can be automatically obtained without manual enumeration of all the possible states.

C. *Estimation for the selection of navigation behaviors*

In order to perform quantitative analysis, the steady-state probability π_i of marking M_i (i=1,2,3,5,6,7,9,10,12,13,14,15, 16) should be obtained from the EMC model in Fig.3.(b). The process to get the steady-state probability π_i is explained in detail in [1] and [7]. The steady-state probability π_i of M_i, means the proportion of time the marking spends in M_i.

The criterion of the behavior selection is "which behavior leads the robot to a goal faster than the other with guaranteeing localization safety." This selection problem can be solved by comparing the frequencies of firing a transition of t_9 and t_{10} of the GSPN model in Fig.2. The physical meanings of t_9 and t_{10} are the completion of a navigation task using *AutoMove* and *Contour tracking*, respectively. Thus, the robot selects the behavior which has higher frequency than the other. The details about the computation of frequency of firing a transition in the analysis of GSPN, i.e., the average number of transition firings in unit time, are explained in [1] and [7]. To summarize, in our system, the frequencies of navigation completion of each behavior can be computed from (1). Therefore, if f_{auto} is larger than f_{cont}, *AutoMove* is selected. Otherwise, *Contour tracking* is chosen.

$$\begin{aligned} f_{Auto} &= \lambda_7 \cdot \pi_1 \\ f_{Cont} &= \lambda_8 \cdot (\pi_2 + \pi_5 + \pi_{10} + \pi_{13}) \end{aligned} \tag{1}$$

The performance estimation, i.e., the calculation of (1), takes place when following occasions arise;

1) At the beginning of a navigation task: Starting a navigation task, the robot plans which behavior is advantageous in a current situation.

2) At the localization state conversion from warning to success: When the localization *warning* occurs, the robot unconditionally executes *Contour tracking* for improving localization accuracy. On its way, if the localizer rediscovers its correct position, the robot should decide whether it remains *Contour tracking* or switches to the *AutoMove*.

D. *Performance analysis*

In addition to the behavior selection, the GSPN model is also used to calculate some performance indices of the system. Typical examples are the utilization of *AutoMove* and *Contour tracking*, the probabilities of localizer *Success* and *Warning*, and the probabilities of path planner *Normal* and *Abnormal* during navigation. They can be obtained from (2), (3), and (4), respectively [1].

Since P_1 and P_2 models the executions of *AutoMove* and *Contour tracking*, the utilization of each of them is the probability when P_1 and P_2 are marked, respectively. Thus, the unitization of each behavior is represented like (2).

$$\begin{aligned} U_{Auto} &= P\{M(P_1)=1\} = \pi_1+\pi_9 \\ U_{Cont} &= P\{M(P_2)=1\} = \pi_2+\pi_5+\pi_{10}+\pi_{13} \end{aligned} \tag{2}$$

$$\begin{aligned} P_{loc,\,succ} &= P\{M(P_5)=1\} = \pi_1+\pi_2+\pi_3+\pi_9+\pi_{10}+\pi_{12}+\pi_{14} \\ P_{loc,\,warn} &= P\{M(P_6)=1\} = \pi_5+\pi_6+\pi_7+\pi_{13}+\pi_{15}+\pi_{16} \end{aligned} \tag{3}$$

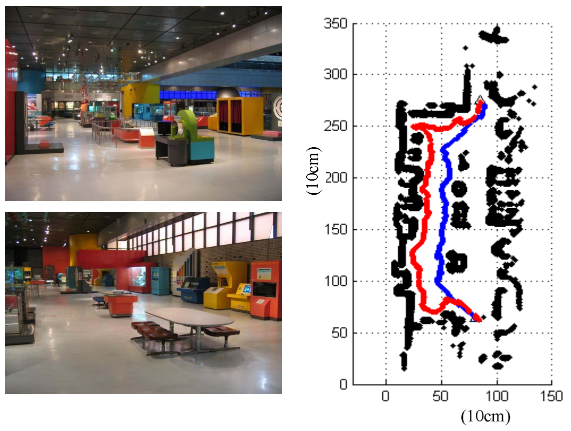
$$\begin{aligned} P_{pp,\,norm} &= P\{M(P_7)=1\} = \pi_1+\pi_2+\pi_3+\pi_5+\pi_6+\pi_7 \\ P_{pp,\,abno} &= P\{M(P_8)=1\} = \pi_9+\pi_{10}+\pi_{12}+\pi_{13}+\pi_{14}+\pi_{15}+\pi_{16} \end{aligned} \tag{4}$$

IV. Experiments and Results

A. *Experimental Environments*

The developed framework is tested on the *Jinny*, which carries out guidance tasks in the *National Science Museum of Korea*. Fig.4.(a) shows a target workspace, one of the sections popular among visitors in the museum. The mission is to navigation from the start point (8.1m, 6.4m) to the goal (8.5m, 27.5m). The size of the grid map (Fig.4.(b)) is about 15m× 35m. As an initial setup, we iterate both *AutoMove* and *Contour tracking* separately in the target workspace five times, and then compute the mean travel time and reference trajectory of each behavior. Fig.4.(b) illustrates each typical trajectory of *AutoMove* and *Contour tracking*. And, it takes 122.33 sec. to complete a navigation task by *AutoMove*, and 186.98 sec. by *Contour tracking* on average.

Although the travel time of *Contour tracking* is 1.53 times as long as that of *AutoMove*, the behavior selection varies with not only the localization warning/recovery rate (λ_3 and λ_4) but also abnormal/recovery rate of the path planner (λ_5 and λ_6).



(a) A target workspace (b) A grid map and reference paths

Fig. 4 Experimental environments.

B. *Behavior selection for 20 runs of navigation*

We run 24 navigation tasks in the target space. The robot successfully reaches to the goal 20 times, and fails the mission 4 times. The major reasons of the failure include a slip for a long time due to an uneven floor, hostile visitors, and a long-term enclosure by visitors. Note that we conduct the experiments without the restrictions on visitors' behaviors.

TABLE III Description of Firing Rates

Rates	Descriptions
λ_1 (λ_2)	λ_1 and λ_2 are firing rates of transitions between *AutoMove* and *Contour tracking* during the localization *Success*. These values are set differently according to which behavior is considered. When the execution rate of *Contour tracking* is computed, λ_1 is set to a large number and λ_2 to a small number. Physical meaning is that when the localizer is in *Success*, *Contour tracking* is selected as possible. Large λ_1 expedites the transition from *AutoMove* to *Contour tracking*, and small λ_2 restrains the reverse transition. Conversely, when *AutoMove* is considered, λ_1 is a small number, and λ_2 is a large number. In conclusion, following fixed values are used; λ_1=0.001, λ_2=1000 for *Automove*, and λ_1 = 1000, λ_2 = 0.001 for *Contour tracking*.
λ_3 (λ_4)	λ_3 is the localization warning rate and λ_4 is the recovery rate. Thus, λ_3 means how frequently the localizer changes its state from *Success* to *Warning* on the average. In other words, its reciprocal indicates the mean of sojourn time of the localizer in *Success* state during navigation. λ_4 has the reverse meaning. These values are updated from the experience after every execution of the mission.
λ_5 (λ_6)	λ_5 and λ_6 are similar to λ_3 and λ_4 except that they are about the path planner. Thus, λ_5 is the abnormal rate of the path planner and λ_6 is the recovery rate. These values are also updated after every execution as does for λ_3 and λ_4.
λ_7 (λ_8)	λ_7 (λ_8) is the rate of task completion using *AutoMove* (*Contour tracking*). These are empirically obtained. After iterating both *AutoMove* and *Contour tracking* in the target workspace several times, the mean travel time and the reference trajectory of *AutoMove* and *Contour tracking* are obtained. When the robot is in initial position, λ_7 and λ_8 are assigned to the reciprocals of mean travel times. In our example, λ_7 = 1./122.33 and λ_8 = 1./186.98. When the robot is in the middle of navigation, λ_7 and λ_8 are estimated from the saved reference trajectories.
λ_9 (λ_{10})	λ_9 (λ_{10}) is the rate of task failure using *AutoMove* (*Contour tracking*). These are estimated from following equation; $\lambda_9 = \lambda_7 \times (N_{fail} / N_{succ})$, $\lambda_{10} = \lambda_8 \times (N_{fail} / N_{succ})$; where, N_{fail}: the number of failures, N_{succ}: the number of successes.
λ_{11}	Since λ_{11} has little effect for performance analysis, it is assigned to a large arbitrary number. In this analysis, the fixed λ_{11} = 1000 is set.

One of the most important modeling issues is how to set the firing rates $\Lambda=\{\lambda_0,\cdots,\lambda_9\}$. These parameters are based on the sources of information shown in Table III.

Table IV shows the changes of collected values λ_3, λ_4, λ_5, and λ_6 during the experiments. They are main parameters which have a great influence on the selection of behaviors. These are gathered from the 20 successful missions. Initially, the set of firing rates Λ is given as $\Lambda=\{0.001, 1000, 0.001, 1000, 0.001, 1000, 0.0082, 0.0053, 0.001, 0.001, 1000.\}$. The localization *Warning* scarcely occurs, and the state is instantly recovered to the *Success* even if it happens. Likewise, λ_5(=1000) and λ_6 (=0.001) are identically assigned to λ_3 and λ_4 to represent that the path planner mostly stay in the *Normal* state. As the number of runs increase, the width of variation of Λ becomes smaller; λ_3, λ_4, λ_5, and λ_6 are obtained by averaging accumulated data.

Fig.5 shows the variation of f_{auto} and f_{cont} during 20 runs of navigation. The system selects the behavior which has the higher frequency of firing a transition than the other. In other words, if f_{auto} is larger than f_{cont}, the *AutoMove* is chosen. The f_{auto} tends to rise as λ_3 decreases or λ_4 decreases. This result is consistent with the fact that it is more likely that *AutoMove* is selected when the localizer remains in the *Success* state longer. Also, f_{auto} increase as λ_5 become smaller or λ_6 becomes larger. It implies that the *Automove* is more advantageous as the path planner sojourns in the *Normal* state longer. In this graph, f_{cont} remain constant to 0.00535 since *Contour tracking* can be used regardless of state changes of the localizer all the time.

The path planner is transited to the *Abnormal* state only if *Path blocking* or *Goal occupation* statuses continue for several seconds to prevent chattering (i.e., switching frequently between the *Normal* and the *Abnormal* state). At twentieth run, λ_5 = 0.0088 and λ_6=0.7451. That is, the sojourn time of the path planner *Normal* and the *Abnormal* is about 113 (≈1/0.0088) sec. and 1.34 (≈1/0.7451) sec., respectively. It reflects that the target environment is extremely dynamic, and thus, the errors do not frequently occur and the recovery follows at once.

TABLE IV Gathered data of some essential parameters

Run	λ_3	λ_4	λ_5	λ_6	f_{auto}	f_{cont}
1	0.0076	0.1546	0.0075	0.3536	0.00763	0.00535
2	0.0083	0.0217	0.0085	0.7072	0.00584	0.00535
3	0.0071	0.0208	0.0073	0.4403	0.00599	0.00535
4	0.0069	0.0251	0.0068	0.5871	0.00634	0.00535
5	0.0077	0.0149	0.0080	0.7339	0.00533	0.00535
6	0.0074	0.0178	0.0078	0.8807	0.00572	0.00535
7	0.0077	0.0166	0.0083	1.0274	0.00554	0.00535
8	0.0080	0.0152	0.0080	1.0406	0.00531	0.00535
9	0.0079	0.0167	0.0080	0.8000	0.00549	0.00535
10	0.0082	0.0178	0.0080	0.8637	0.00554	0.00535
11	0.0082	0.0187	0.0078	0.7983	0.00563	0.00535
12	0.0085	0.0165	0.0085	0.8009	0.00534	0.00535
13	0.0086	0.0149	0.0082	0.8677	0.00513	0.00535
14	0.0087	0.0134	0.0088	0.8154	0.00490	0.00535
15	0.0086	0.0158	0.0087	0.7893	0.00524	0.00535
16	0.0087	0.0154	0.0088	0.7923	0.00517	0.00535
17	0.0087	0.0154	0.0090	0.6988	0.00516	0.00535
18	0.0087	0.0152	0.0091	0.6706	0.00513	0.00535
19	0.0085	0.0156	0.0089	0.7078	0.00523	0.00535
20	0.0083	0.0163	0.0088	0.7451	0.00535	0.00535

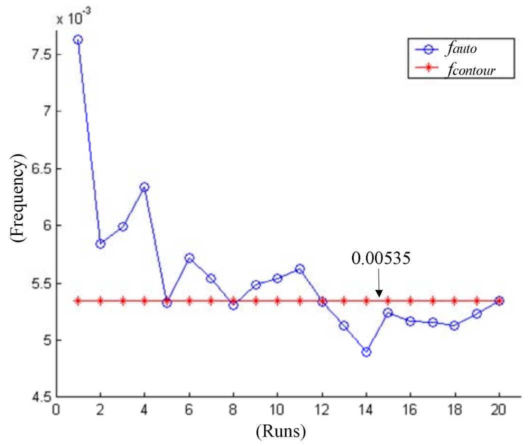
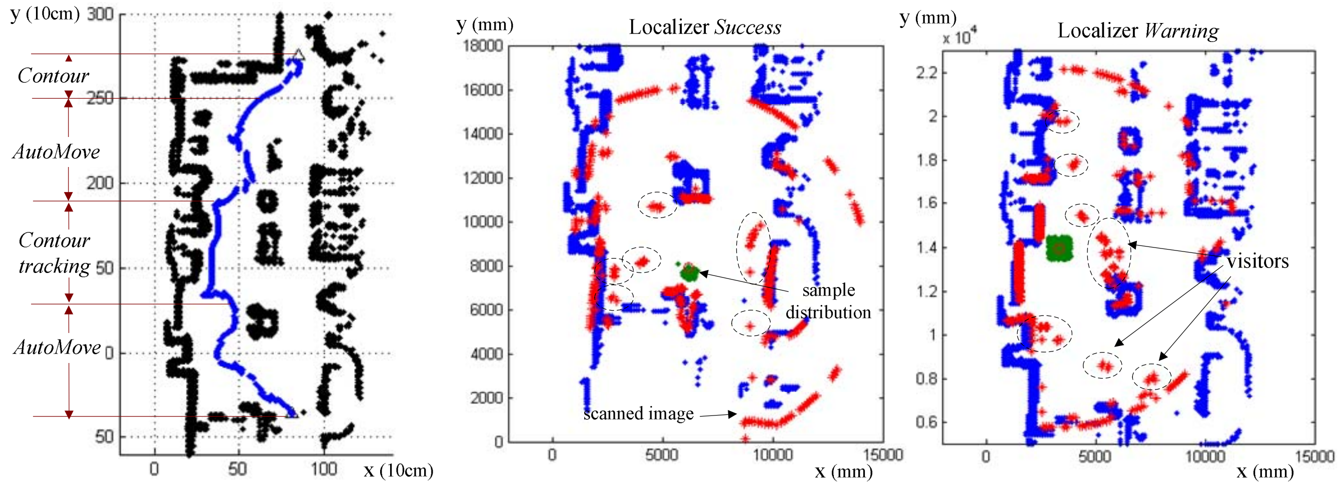


Fig. 5 Variation of f_{auto} and f_{cont} during 20 runs of navigation.



(a) A trajectory and changes between behaviors (b) Localization data during *AutoMove* (c) Localization data during *Contour tracking*

Fig. 6 Experimental results during one execution of the navigation task.

C. *Behavior selection during a single navigation*

The above results only show the performance analysis after the completion of the tasks. Hence, they do not describe the changes of behaviors during a single navigation according to the dynamic properties of an environment. Fig.6 shows the results of the behavior estimations in this case. Although the robot initially starts with *AutoMove*, the robot changes its motion to *Contour tracking* when the localization *Warning* is detected. On its way, if the localizer is recovered to the *Success* state, the robot carries out the behavior estimation, (i.e., the calculation of (1)) again to decide whether it remains *Contour tracking* or changes to the *AutoMove*. In this example, the robot shifts to the *AutoMove*. Fig.6.(a) shows the final robot's trajectory. Fig.6.(b) and Fig.6.(c) illustrate the localization results during the *AutoMove* and the *Contour tracking*. It contains the information about the local map, laser scan data, sample distributions, and estimated position each calculation. Fig.6.(b) is a typical example of the localizer *Success*, while Fig.6.(c) is an instance of the localizer *Warning*. As shown in these pictures, the environment is very crowded and dynamic by visitors.

D. *Performance analysis model*

As the example of navigation performance analysis, we present the experimental results of the probabilities of localization *Success* and *Warning* derived from (3). Fig.7 shows the variation of $P_{loc,succ}$ and $P_{loc,warn}$ during 20 runs of experiments. Note that $P_{loc,succ}+P_{loc,warn}=1$, and naturally, $P_{loc,succ}$ decreases with the increase of $P_{loc,warn}$.

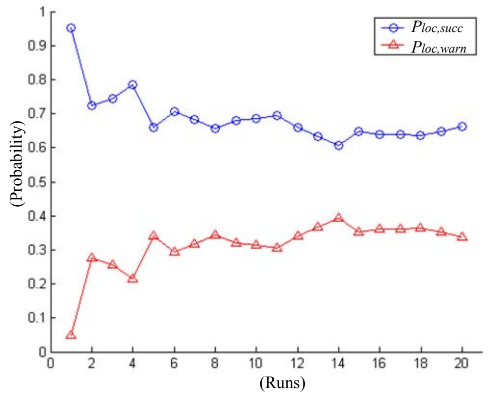


Fig. 7 Variation of $P_{loc,succ}$ and $P_{loc,warn}$ during 20 runs of navigation.

V. Conclusion

This paper presents a selection framework of multiple navigation behaviors for a service robot based on GSPN formalism. Modeling, analysis, and performance evaluation are conducted on firm mathematical foundation. Although, we consider simplified model which considers only two navigation behaviors and two components, the localizer and the path planner, it does not undermine the advantages of the proposed approach. . Rather, the more complicated the model is, the more advantageous our approach is. We can obtain a new model without complexity explosion.

We successfully conduct experiments in our target workspace, the *National Science Museum of Korea*. The results clearly show that the proposed strategy is feasible and useful to select an appropriate navigation behavior in a real environment.

References

[1] Gunhee Kim, Woojin Chung, and Munsang Kim, "A Selection Framework of Multiple Navigation Primitives Using Generalized Stochastic Petri Nets", in Proc. of the *IEEE Int. Conf. on Robotics and Automation*, Barcelona, Spain, April 18-22, 2005.

[2] Gunhee Kim, Woojin Chung, Sangmok Han, Kyung-Rock Kim, Munsang Kim, and Richard H. Shinn, "The Autonomous Tour-Guide Robot Jinny", in Proc. of the *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[3] Gunhee Kim, Woojin Chung, Munsang Kim, and Chongwon Lee, "Tripodal Schematic Design of the Control Architecture for the Service Robot PSR," in Proc. of the *IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, pp.2792-2797, 2003.

[4] Gunhee Kim, Woojin Chung, Munsang Kim, and Chongwon Lee, "Implementation of Multi-Functional Service Robots Using Tripodal Schematic Control Architecture," in Proc. of the *IEEE Int. Conf. on Robotics and Automation*, New Orleans, USA, pp.4005-4010, 2004.

[5] Woojin Chung, Gunhee Kim, Munsang Kim, and Chongwon Lee, "Integrated Navigation System for Indoor Service Robots in Large-scale Environments," in Proc. of the *IEEE Int. Conf. on Robotics and Automation*, New Orleans, USA, pp.5099-5104, 2004.

[6] Dongheui Lee, Woojin Chung, Munsang Kim, "A Reliable Position Estimation Method of the Service Robot by Map Matching," in Proc. of *the IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, 2003.

[7] Jiacun Wang, *Timed Petri Nets Theory and Application*, Norwell, MA: Kluwer Academic Publishers, 1998.

[8] MengChu Zhou and Mu Der Jeng, "Modeling, Analysis, Simulation, Scheduling, and Control of Semiconductor Manufacturing Systems: A Petri Net Approach," *IEEE Trans. Semiconductor Manufacturing*, vol. 11, no. 3, pp.333-357, August 1998.

[9] R.Y. Al-Jaar and A.A. Desrochers, "Performance Evaluation of Automated Manufacturing Systems using Generalized Stochastic Petri Nets," *IEEE Trans. Robotics and Automation*, vol. 6, no. 6, pp.621-639, December 1990.

[10]J. Diard, P. Bessiere, and E.Mazer, "A theoretical comparison of probabilistic and biomimetic models of mobile robot navigation," in Proc. of the *IEEE Int. Conf. on Robotics and Automation*, New Orleans, USA, pp.933-938, 2004.

[11]P. Althaus, H. Ishiguro, T. Kanda, T. Miyashita, and H.I. Christensen, "Navigation for Human-Robot Interaction Tasks," in Proc. of the *IEEE Int. Conf. on Robotics and Automation*, New Orleans, USA, pp.1894-1900, 2004.

[12]G. Booch, J. Rumbaugh, I. Jacobsen, *Unified Modeling Language User Guide*, Addison Wesley, Longman, 1997.

[13]S. Koenig and R.G. Simmons, "Xavier: A Robot Navigation Architecture Based on Partially observable Markov Decision process Models," *Artificial Intelligence and Mobile Robots*, D. Kortenkamp, R.P. Bonasso, and R. Murphy Eds.: AAAI Press, 1998, pp. 91-122.

[14]N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal Navigation- Mobile Robot Navigation with Uncertainty in Dynamic Environments," in Proc. of the *IEEE Int. Conf. on Robotics and Automation*, 1999.

[15]K. Albert, K. Jensen, and R. Shapiro, "DESIGN/CPN: A tool package supporting the use of colored nets," *Petri Net Newsletter*, No. 32, p. 22–35. Available: http://www.daimi.au.dk/designCPN/

[16]R. German, C. Kelling, A. Zimmermann, and G. Hommel, "TimeNET: a toolkit for evaluating non-Markovian stochastic Petri nets," *J. Performance Evaluation*, Vol. 24, pp. 69-87, 1995. Available: http://pdv.cs.tu-berlin.de/~timenet/

[17]Kurt Konolige, "A Gradient Method for Realtime Robot Control," in Proc. of the *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Japan, pp.639-646, 2000.