Design of Navigation Behaviors and the Selection Framework with Generalized Stochastic Petri Nets toward dependable navigation of a mobile robot

Chang-bae Moon and Woojin Chung, Member, IEEE

Abstract—There have been two major streams for the motion control of mobile robots. The first is the model-based deliberate control and the second is the sensor-based reactive control. Since two schemes have complementary advantages and disadvantages, one cannot completely replace the other. There are a variety of environmental conditions which affect the navigation performances. The main idea of this paper is to design discrete navigation behaviors and to integrate behaviors by an appropriate selection framework. In this paper, we propose a behavior selection framework using the GSPN (Generalized Stochastic Petri Nets). We have designed two navigation behaviors which show completely different performances. In order to define behavior selection criteria, two kinds of navigation statuses are defined to monitor navigation performances of the robot. The proposed navigation strategy is simulated using the open source simulator Player/Stage to investigate the performances in a variety of conditions. Through the simulations, it was made clear that different behaviors show remarkably different performances. Moreover, the average navigation time of the proposed behavior selection framework is significantly decreased than that of any single navigation scheme DWA or tracking.

I. INTRODUCTION

In the past two decades, a variety of controllers have been developed for the autonomous navigation of mobile robots. There have been two major streams for the motion control of mobile robots. The first is the model-based deliberate control and the second is the sensor-based reactive control, as introduced in [1]. The model-based navigation schemes compute the motion command from the environmental model. Although the model-based schemes show optimal performances in static environments, it is not recommended in highly dynamic environments. The sensor-based reactive schemes compute the motion command by using the sensor information directly. The reactive schemes are suitable for highly dynamic environments. However, the reactive schemes have some drawbacks such as local trap situations. Since two schemes have complementary advantages and disadvantages, one cannot completely replace the other.

There have been many studies on model-based schemes, sensor-based and the combination of the two, for example, in [2]. Brock and Khatib presented the Global Dynamic Window Approach [3] which integrates a well-known reactive scheme DWA (Dynamic Window Approach) [4] and path planner which computes the NF (Navigation Function) [5]. Ulrich and Borenstien proposed the VFH* (Vector Field Histogram*) in [6]. The VFH* uses a look-ahead verification to analyze the sequences of heading direction. The detailed overview on the integration approaches using reactive schemes was presented in [2]. Although these approaches can handle the local minimum problem of the reactive schemes, they still have drawbacks as mentioned in [7] and [8]. Stachniss and Burgard presented a method which integrates path planning with sensor-based collision avoidance in [7].

The other approaches exploiting multiple navigation techniques are presented in [9]-[11]. Borenstein and Koren presented a method to deal with the local minimum and cyclic trap state in [9] using the path monitor which monitors the movement of a robot. The control schemes presented in [10] and [11] are adopted for reflexive navigation behaviors such as obstacle avoidance or goal approaching behaviors. The presented method in [10] and [11] are highly dependent on high-level planner’s ability. Hence, the most challenging problem is the design of the interaction model between the deliberative layer and reactive layer as introduced in [12].

The mobile robot control schemes were proposed in [13] and [14] using the Petri-Net formalism to deal with multi-robot coordination. We proposed a behavior selection framework using the GSPN in [15]. The basics of GSPN are introduced in [16]. In [16], we designed the framework using model-based behavior and wall-following behavior considering the localizer reliability. In [17], we proposed a selection framework for multiple navigation tasks. In our previous works, we exploited a wall-following behavior to deal with the localizer failure.

Our previous works was focused to deal with error-recovery. However, the quantitative comparison was not sufficiently carried out in our prior works. In this paper, we exploit multiple motion controllers and quantitative comparison of motion controllers was carried out using the multi robot simulator. Moreover, the navigational performance was improved by accumulating stochastically the previous navigation experiences.

There are a variety of environmental conditions which affect the navigation performances. The main idea of this
paper is to design discrete navigation behaviors and to integrate behaviors by an appropriate selection framework. We have designed two navigation behaviors which show completely different performances. In order to define behavior selection criteria, two kinds of navigation statuses are defined to monitor navigation performances the robot. We will present a stochastic modeling method to estimate the navigation behavior’s internal states under the consideration of the overall performance.

The remaining parts of this paper are organized as follows. In section II, we briefly explain the two navigation behaviors that are exploited in this paper. In section III, we present the behavior selection framework using the GSPN. The simulation results are presented in section IV. Finally, the concluding remarks are presented in section V.

II. NAVIGATION BEHAVIORS

A. The sensor-based behavior Dynamic Window Approach

The sensor-based reactive behavior DWA is exploited to deal with highly dynamic environments. However, the DWA is not recommended for entering a doorway. In addition, U-shaped obstacles may lead a robot to local trap situations.

Since the DWA is a local motion controller, it is recommended to be used together with a global path planner for generating way-points. In this paper, we exploit the gradient method [19] proposed by Konolige. The waypoints are generated by segmenting the gradient path into waypoint distance, $d_w$. If there are invisible adjacent waypoints, a waypoint is inserted in the invisible region. The visibility between each adjacent waypoint is recursively checked until there is no invisible adjacent waypoint.

By using this algorithm for generating waypoints, adjacent waypoints can be connected by a straight line. This waypoint-generation algorithm is advantageous since the risk of the local-minimum problem can be decreased.

B. The model-based behavior Trajectory Tracking

The model-based behavior trajectory tracking is advantageous because it follows the optimal path. However, the tracking behavior has limitations as follows. The tracking behavior is not recommended in highly dynamic environments because it requires frequent re-planning of the trajectory because the planned path can be blocked by moving obstacles. Moreover, the tracking behavior requires high precision of the localizer.

The tracking behavior is composed of the path-planner, trajectory generator and tracking controller. The tracking behavior exploits the gradient path planner for generating a collision free path. By the use of the Bubble-Bands algorithm [20], we generated a Cubic B-Spline curve in order to obtain smooth curve which satisfies nonholonomic constraints.

$$\kappa(s) = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{3/2}} \quad (1)$$

$$v = \begin{cases} \frac{v_{\text{max}}}{\omega_{\text{max}}} ; & \kappa(s) \leq \omega_{\text{max}} / v_{\text{max}} \\ \frac{\omega_{\text{max}}}{\kappa(s)} ; & \kappa(s) > \omega_{\text{max}} / v_{\text{max}} \end{cases} \quad (2)$$

The tracking trajectory $\Gamma = \{x,y,\theta,v,\omega\}$ is computed by considering the limit of the radius of curvature of the robot. Using (1), we compute the radius of curvature, $k(s)$. The translational velocity is computed by using (2). We exploit Kanayama’s tracking algorithm in [21] for tracking the trajectory.

III. NAVIGATION BEHAVIOR SELECTION FRAMEWORK USING GSPNs

A. Design of a Navigation Framework using GSPNs

The GSPN is extension of the Stochastic Petri Nets. The advantages of the GSPN for modeling a navigation system are as follows. The Finite State Automata used in [22] and [23] are incapable of describing the concurrency of a system. The GSPN describes the concurrency of sub-system’s state individually using the token representation. The qualitative and quantitative analysis ability is superior to the Finite State Automata. The number of places and transitions of GSPN increases linearly as introduced in [24]. However, the number of states in the Partially Observable Markov Decision Process used in [25] increases exponentially. In [15], the advantages of the GSPN were presented in detail.

Fig. 1 shows the designed GSPN model using two behaviors that are tracking and DWA. A detailed description of places and transitions is presented in Table I. Transitions that have constant values of firing rate are shown in Table I. For the remaining transitions, the firing rate is updated after the completion of each navigation task. By firing the timed transition, $T0$, the navigation task is started. The performance evaluation of each behavior using the GSPN is carried out when a token exists in the place, $P1$. The performance evaluation is carried out considering the expected navigation completion time and failure rate.

Suppose that the robot is moving in static environment. Then, the tracking behavior is preferred because it is advantageous to move along the optimal path. If the robot encounters highly dynamic environment, the path planner frequently updates collision free paths. If the path deformation takes place too frequently, then it can be concluded that the robot is in a dynamic environment. From the viewpoint of environmental status, the planner warning state is activated. As a result, the robot possibly switches its behavior into DWA to survive in the dynamic environment. The transition reactive warning ($T7$) is fired when the robot falls in local trapped situations. This control logic is modeled in Fig. 1.

If the transition planner warning ($T5$) event is fired when the navigation behavior is tracking ($P2$), the planner status is
changed from the planner normal state (P4) to the planner warning state (P5). The result of firing T5 is that the current navigation behavior tracking is changed into the DWA by firing T3.

![Fig. 1. A GSPN Model with two navigation behaviors and two internal statuses.](image)

Table I

<table>
<thead>
<tr>
<th>Place Description</th>
<th>Firing Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0 Navigation ready</td>
<td>λ0 = 1.0</td>
</tr>
<tr>
<td>P1 Navigation behavior selection</td>
<td>λ1</td>
</tr>
<tr>
<td>P2(P3) Tracking / DWA</td>
<td>λ2</td>
</tr>
<tr>
<td>P4(P5) Planner normal (warning)</td>
<td>λ4</td>
</tr>
<tr>
<td>P6(P7) Reactive normal (warning)</td>
<td>λ6</td>
</tr>
<tr>
<td>P8(P9) Navigation success (failure)</td>
<td>λ8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Timed Transition</th>
<th>Description</th>
<th>Firing Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0 Start navigation</td>
<td>λ0 = 1.0</td>
<td></td>
</tr>
<tr>
<td>T1(T2) Tracking(DWA) selected by the performance estimation</td>
<td>λ1(λ2)</td>
<td></td>
</tr>
<tr>
<td>T5(T6) Planner warning (recovery)</td>
<td>λ5(λ6)</td>
<td></td>
</tr>
<tr>
<td>T7(T8) Reactive warning (recovery)</td>
<td>λ7(λ8)</td>
<td></td>
</tr>
<tr>
<td>Ta6(Ta8) Planner (reactive) warning is recovered when the navigation behavior is tracking (DWA)</td>
<td>λa(λa)</td>
<td></td>
</tr>
<tr>
<td>T9(T10) Navigation completed by tracking (DWA)</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>T11(T12) Navigation failure under the tracking (DWA)</td>
<td>λ1(λ2)</td>
<td></td>
</tr>
</tbody>
</table>

For the performance evaluations, some definitions are made as follows:

\[ v_e = \text{dist}_{opt} / t_{\text{navi}} \]  
\[ \lambda_1 = \frac{v_e}{\text{dist}_{opt}} \]  
\[ t_{\text{fail}} = \text{dist}_{opt} / v_{\text{fail}} \]

The rates of success and failure of navigation are computed as follows. The empirical velocity \( v_e \) is computed by using (3). \( \text{dist}_{opt} \) is the shortest path length and \( t_{\text{navi}} \) is the overall time for navigation. The empirical velocity refers to the intrinsic velocity of each behavior on the basis of experimental results. The firing rates of firing of navigational success, \( \lambda_s(\lambda_a) \), are computed from (4). Equation (4) converts the empirical velocity into GSPN’s firing rates (s^{-1}) using the computed shortest path length. We should make a criterion to measure the navigation failure quantitatively. The navigation failure time \( t_{\text{fail}} \) is computed by using (5). \( v_{\text{fail}} \) is the failure velocity. If the total navigation time is longer than the failure time \( t_{\text{fail}} \), the navigation is considered to be a failure and the completion time of the failed navigation task is recorded as \( t_{\text{navi,total}} \).

\[ t_{\text{navi,total}} = t_{\text{navi,tracking}} + t_{\text{navi,dwa}} \]
\[ \lambda_{11} = \frac{t_{\text{navi,tracking}}}{t_{\text{navi,total}}} \]
\[ \lambda_{12} = \frac{t_{\text{navi,dwa}}}{t_{\text{navi,total}}} \]  

The failure rates of each behavior are required for updating \( \lambda_{11} \) and \( \lambda_{12} \). However, the computation of the navigation failure rates for each behavior is not straightforward because the GSPN framework exploits multiple navigation behaviors. In this paper, we compute the failure rates using (6). The rates are normalized by the total navigation time in proportion to navigation time of each behavior. \( t_{\text{navi,total}} \) is the total navigation time. \( t_{\text{navi,tracking}} \) is the time used for the tracking behavior, and \( t_{\text{navi,dwa}} \) is the time used for DWA behavior.

### B. Performance Estimation using the GSPN

The performance estimation for each behavior is carried out by computing the throughput of the navigation success transitions of that behavior, i.e., \( T9 \) (resp., \( T10 \)) for tracking (resp., DWA). The throughput is computed by multiplying the firing rate and the probability of the steady state that is related to the transition. The method of estimating the performance using the throughput is introduced in [15], [16]. The throughputs of the navigation behaviors are decreased when the probability of the sub-system’s warning states is increased. The physical meaning of the throughput is the navigation success frequency (s^{-1}). For instance, if the throughput of the tracking behavior \( T9 \) is higher than that of the DWA behavior \( T10 \), the expected navigation time of tracking behavior is shorter than that of the DWA. In summary, if the throughput value of \( T9 \) is larger than that of the throughput of \( T10 \), the tracking behavior is selected; otherwise, the DWA is selected.

In this study, we monitor the firing rates of the events, \( T1 \) and \( T2 \), which imply the preference of the tracking and DWA behavior respectively. The throughput is computed by using the firing rates for tracking \( \lambda_s \) and DWA \( \lambda_a \) with same parameters to compute each behavior’s performance estimation. As a result, the performance estimation is carried out in a single step through this computational method.
C. State Estimation of the Sub-systems

We have to design a model for switching the navigation behavior’s internal states. If the switching model is not appropriately designed, the risk of chattering problem may increase. The previous research presented in [25] was based on the environmental shapes such as corridor and U-shaped. Since our selection model adopts discrete navigation behaviors, the navigation behavior’s internal state transition model should be designed differently. For example, if the DWA behavior is selected, the path-regeneration does not take places. As a result, different criteria should be defined for estimating each navigation behavior’s internal states.

We record the count of the instances of path planning as $N_p$ in a constant time-interval, $T_v$, for monitoring the state of the tracking behavior. The state of the DWA behavior is estimated through monitoring the velocity to a given waypoint. The state estimation of DWA behavior is carried out by counting the number, $N_r$, when the velocity is lower than $V_{low}$, in the same manner as that of the tracking behavior. Although it is easy to obtain $N_r$, the decision of threshold is another significant problem.

By exploiting the Poisson model [26], the counting process can be modeled directly into transition probability because the Poisson model explicitly includes the counted numbers. Moreover, the Poisson model is approximated to the exponential density which used to the timed transition of the GSPN by Poisson limit theorem [26].

The transition probabilities should be computed by considering the overall system states. For a mixed probabilistic model of transition states, it is advantageous to use the GSPN because the GSPN supports tools for analyzing stochastic state probabilities such as the token probability. In our GSPN model, the token probability simply means the state probability for each sub-system’s internal place because each sub-system is modeled to have one token for one sub-system.

$$N_f = \arg \left\{ F_p \left\{ N \left| \lambda_j, T_k \right. \right\} \geq \Pr \{ P_i = 1 \} \right\} \quad (7)$$

$$p(N \left| \lambda_j, T_k \right.) = \frac{(\lambda_j T_k)^N}{N!} \exp(-\lambda_j T_k) = \frac{\lambda_j T_k}{N} \cdot P(N-1 \left| \lambda_j, T_k \right.) \quad (8)$$

The planner and reactive warning events are $N \sim \text{Poisson}(\lambda_j T_k)$. The transition threshold value, $N_f$, is computed using (7). $\Pr \{ P_i \}$ is the token probability of $P_n$, e.g., $P5$ for the planner warning state. The firing conditions for the planner and reactive warning events are $N_p > N_f$ and $N_r > N_f$, respectively. $F_p \left\{ t \right\}$ is the cumulative Poisson distribution function with frequency $\lambda_j$, e.g., $\lambda_r$ for the case of planner warning. The analytic cumulative density function of the Poisson probability density function includes the $\Gamma$-function, which entails high computational burden. Hence, in this study, we compute the cumulative density function by integrating the probability using (8) recursively.

$$t_f = \arg \left\{ F_p \left\{ t \left| \lambda_j \right. \right\} \geq \Pr \{ P_i = 1 \} \right\} + t_a \quad (9)$$

$$F_p \left\{ t \left| \lambda_j \right. \right\} = \begin{cases} 1 - e^{-\lambda_j t} & ; t \geq 0 \\ 0 & ; t < 0 \end{cases} \quad (10)$$

The planner and reactive recovery events are fired after the time $t_f$ is computed using (9) from the time when the warning event is fired. The recovery events are $t \sim \text{Exp}(\lambda_r \cdot t_a \cdot$ required time for adjusting a robot’s heading direction. $F_p \left\{ t \right\}$ is the cumulative density function of the exponential probability density with $\lambda_r$ being computed using (10).

IV. SIMULATION RESULTS

A. Simulation Settings

The simulation is carried out using the Player/Stage tool [19]. The Player robot client and server are widely used robot control interfaces, which abstract the connection between the navigation software and the real/simulated robot. The Stage simulator simulates a population of mobile robots that work in a two-dimensional environment.

Fig. 2. Simulation environment. (a) A real environment, (b) Stage world model (obstacles are denoted by blue dots).

Fig. 2(a) shows a practical environment. We generate a simplified simulation environment based on Fig. 2(a). Fig. 2(b) shows a stage environment. The simulated environment is 43m wide and, 22m long. The doorway is 1m wide in accordance with CAD map of the office building. The diameter of the simulated mobile robot is 0.5m and the mobile robot is equipped with a SICK laser scanner. The maximum translational velocity was set to 0.5m/s. The speed of moving obstacles is randomly selected between 0.8~1.4m/s, which refers to the average speed of human walking.
The empirical velocity, $v_e$, is obtained from 20 replications of the navigational results between G1 and G8 in the absence of moving obstacles. The empirical velocity is 0.28 m/s for DWA and 0.35 m/s for tracking. The initial frequency of the GSPN is set to $\lambda_1 \sim \lambda_2 = 0.5$, then $\lambda_1, \lambda_2 = 0.01$.

**B. Comparisons of Navigation Behaviors**

The simulation scenario of Case1 is that the mobile robot visits goals G1, G2, G3... G7 sequentially start from G7 without dynamic obstacles. The simulation, Case1, is carried out to measure the performance in complex and static environments. The simulation scenario of Case2 is that the mobile robot between two goal points G1 and G8 with 20 moving obstacles. The Case2 is carried out to measure the performance in dynamic environments. Each simulation is carried out 28 times.

![Fig. 3. Simulation results of the Case1 and Case2.](image)

<table>
<thead>
<tr>
<th></th>
<th>DWA</th>
<th>Tracking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case1</td>
<td>Mean Time(s)</td>
<td>185.7</td>
</tr>
<tr>
<td></td>
<td>Std. Dev.</td>
<td>126.7</td>
</tr>
<tr>
<td></td>
<td>Success (%)</td>
<td>82</td>
</tr>
<tr>
<td>Case2</td>
<td>Mean Time(s)</td>
<td>105.8</td>
</tr>
<tr>
<td></td>
<td>Std. Dev.</td>
<td>16.3</td>
</tr>
<tr>
<td></td>
<td>Success (%)</td>
<td>100</td>
</tr>
</tbody>
</table>

Fig. 3 shows the simulation results for Case1 and Case2. The simulation results are listed in Table II. The time of navigation failure is measured as $t_{fail}$ to reflect the risk of failure. In complex and static environments such as Case1, the average navigation time of the tracking behavior is 38% shorter than the case of the DWA. However, in a dynamic environment, Case2, the performance of the tracking behavior is significantly inferior to that of the DWA.

The performance of the DWA is decreased when the robot was entering a narrow doorway. The navigation failure of the DWA took place when the robot was stuck in the local minimum situation. In the static environment, the navigation failure of the tracking behavior is due to the path planning failure by the localization error around a narrow doorway. The performance of the tracking behavior is significantly decreased due to the frequent update of the path-planning because moving obstacles block the generated path. The simulation results of Case1 and Case2 clearly show that the navigation schemes show completely different performances and both navigation schemes are required in practical environments.

**C. Simulation results using the proposed navigation framework**

The simulation scenario of Case3 is that the mobile robot visits goals G1, G2, G3... G7 sequentially start from G7 with 20 moving obstacles. The Case3 is carried out to measure the performance in complex and dynamic environments.

![Fig. 4. Simulation results of the Case3.](image)

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Mean Time(s)</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWA</td>
<td>207.3</td>
<td>128.2</td>
</tr>
<tr>
<td>Tracking</td>
<td>223.1</td>
<td>92.4</td>
</tr>
<tr>
<td>GSPN</td>
<td>117.8</td>
<td>28.0</td>
</tr>
</tbody>
</table>

Fig. 4 shows the simulation results for Case3. The simulation results are summarized in Table III. As shown in Fig. 4, the proposed scheme for the selection of navigation behavior through the GSPN shows superior performance to a single navigation scheme. The average navigation time is shorter than the DWA by 43% and tracking by 47%. Moreover, the navigation success rate of the proposed scheme is 100%, which means that the navigation time for carrying out the navigation task is always smaller than the failure time, $t_{fail}$.

![Fig. 5. GSPN Trajectory from G1 to G2 of simulation Case3.](image)

Fig. 5 shows one instance of the navigation results for Case3 from G1 to G2. The mobile robot starts its navigation task by using the DWA behavior at the starting position G1, because the computed throughput of the DWA ($=0.43 \times 10^2$) is higher than that of the tracking ($=0.40 \times 10^2$). While the robot moving around the lobby, it is surrounded by the moving obstacles at point A and the reactive warning event is fired. As a result, the behavior is changed to tracking. By using the tracking behavior, the robot moves about 1.2 m. There are several obstacles in the lobby area. As a result, the planner warning event is fired at point B. Then, the DWA...
behavior is selected by computing the navigation behavior’s performance. The computed throughputs of the DWA ($=0.51\times10^2$) is higher than that of the tracking ($=0.46\times10^2$). Around the doorway, the reactive warning event is fired at point C. Finally, the mobile robot carries out its remaining navigation task in 18s by using tracking. Since the computed recovery time was 63.41s, the reactive recovery event does not fire.

Fig. 6. Variation of the Case1 firing rates.

Fig. 6 shows the variation of the firing rates, $\lambda_1$ and $\lambda_2$, in Case1 and Case3. The firing rates $\lambda_1$ and $\lambda_2$ imply the preference of the tracking and DWA behavior respectively. When a navigation task was completed, the firing rates $\lambda_1$ and $\lambda_2$ are updated by using the each behavior’s selection rate of the completed navigation task. As shown in Fig. 6, $\lambda_1$ is significantly larger than $\lambda_2$ after the execution of eight navigation tasks in a static environment, Case1. This result implies that the tracking behavior is mostly selected in environments of the type, the Case1.

V. CONCLUSION

In this paper, we proposed a behavior selection framework using the GSPN. We carried out the performance estimation of two navigation behaviors DWA and tracking. The simulation results make it clear that the navigation schemes show complementary advantages and disadvantages. The average navigation time of the proposed behavior selection framework is shorter than that of the DWA about 43% and the tracking about 47%. Moreover, the navigation success rate of the proposed scheme is 100%. The simulation results show that our behavior selection framework is more efficient and robust than using any single navigation scheme.

REFERENCES


