【FRC3-2】

39th International Symposium on Robotics 2008
Seoul, Korea / October 15~17, 2008

Redesign and Analysis of GSPN-based Navigation Selection Framework for an Indoor Mobile Service Robot

Chang-bae Moon, Woojin Chung*, and Jae-Bok Song
Division of Mechanical Engineering
Korea University
Anam-Dong, Seongbuk-Gu, Seoul 136-701, Korea
E-mail : <lunar97, smartrobot, jbsong>@korea.ac.kr

Abstract

In this paper, we carry out the experimental comparison between the *Tracking* and *AutMove* behavior. The experimental results show that the two navigation schemes have appropriate environments. Therefore, we redesign the navigation framework proposed method in [1]. As pointed out in [1], the main advantage of the GSPN is that the numbers of places and transitions only increase linearly as the system complexity increase, whereas the number of states in the MPs increase exponentially. The redesigned model in this paper, EMC (Embedded Markov Chain) matrix has 36x36 dimensions. The system modeling using the GSPN is quite useful, even though the state-space increases exponentially. Also, the performance analysis can be done automatically using mathematical formulation of the GSPN. The experimental results show that the selection between the *Tracking* and *Automove* by added components increases the navigation performance.

1. Introduction

Mobile robot navigation approaches can be classified into reactive control scheme and model-based scheme. A Reactive control scheme is appropriate in highly uncertain environments since it does not affected by the pre-computed results. However, the reactive control scheme does not show globally optimal navigation results. On the other hand, a model-based scheme generally shows high performance in static environments. However, model-based approaches generally show low reactivity. Therefore, a navigation framework which utilizes the both navigation scheme with a appropriate coordination is desirable.

We proposed a formal selection framework of multiple navigation behaviors for a service robot. [1] In [1], the model-based motion control *AutoMove* and sensor-based reactive control *Contour-tracking* behaviors are used for selection framework. Based on the GSPN (Generalized Stochastic Petri-Nets), the dynamic modeling and static modeling was done. Also, it exploits the localizer status and path-planner status to estimate the navigation behavior performance. The experimental results show that the behavior selection scheme is useful in human co-existing real environments.

In [7], we redesigned the navigation behavior *AutoMove* by using DWA (Dynamic Windows Approach) with the global path planner. Also, the *contour-tracking* is redesigned using the CVM (Curvature Velocity Method) which does not compute the heading cost. In [7], experiments were carried out in a large office building. The experimental results show that a robot can navigate robustly even the environmental changes and efficiently with *AutoMove*.

AutoMove shows high speed in general environments. However, the tracking which used in [1] shows safe and high-speed navigation results in static cluttered environments. Since the *AutoMove* exploit the sensor-based myopic approach, it does not carry out pre-planning; it cannot slow down sufficiently its translational velocity.

In [1], we had claimed that even the proposed navigation framework considered just two navigation behaviors and the localizer and path planner this simplified assumption does not undermine the advantages of the proposed framework. As the model becomes more complex, the proposed approach becomes clear because the proposed navigation framework supports the modular and incremental design.

In this paper, we add a *tracking* behavior. Also DWA which used as an *AutoMove* and *Contour-tracking* is still used. Furthermore, new places such as narrow area and normal area and transitions are added related with *tracking* behavior and area evaluation component. As a result, the full state space of the model is grown to 48 states including vanishing states and tangible state which consider only timed transition without immediate transition. The markov transition probability matrix has 2304 elements. It is clear that without the mathematical framework of the GSPN, it is too hard to model the system using MPs (Markov processes) or FSA (Finite State Automata) model directly.

This paper organized as follows. In section 2, we introduce our navigation behaviors. Furthermore, the newly added tracking behavior and comparison results will be presented. In section 3, a modified navigation framework and its analysis will be presented. Simulation results are presented in section 4. Finally, the concluding remarks are presented in section 5.

2. Navigation behaviors

2.1. AutoMove behavior

The *automove* behavior is designed to deal with most normal situation which relatively small difference with the previous given map with dynamic obstacle avoidance capability. We basically designed the motion controller based on the DWA (Dynamic Window Approach).

AutoMove behavior uses the follows shortest path generated by the Gradient method. In [1], *AutoMove* behavior follows shortest path using the Gradient method. However, we found out that the shortest path is dangerous in many cases, because the robot approaches the region which is too close to obstacle boundaries. On the other hand, it is quite different from human's walking patterns therefore many people may feel that the robot's movement is somewhat strange. In order to overcome those problems, we exploit the isovist.

The isovist is used to expect a human walking pattern in a large building using the visibility information. The isovist means that the area in spatial environment directly visible within the space. In this paper, the computation of the isovist is carried out using the ray-casting method without the range limitation. We can compute the visible area using the sum of the ray-casting distance. The detailed computation method of the isovist and experimental results are shown in [3]. So far, we use the isovist map that is computed offline because of the long computing time.

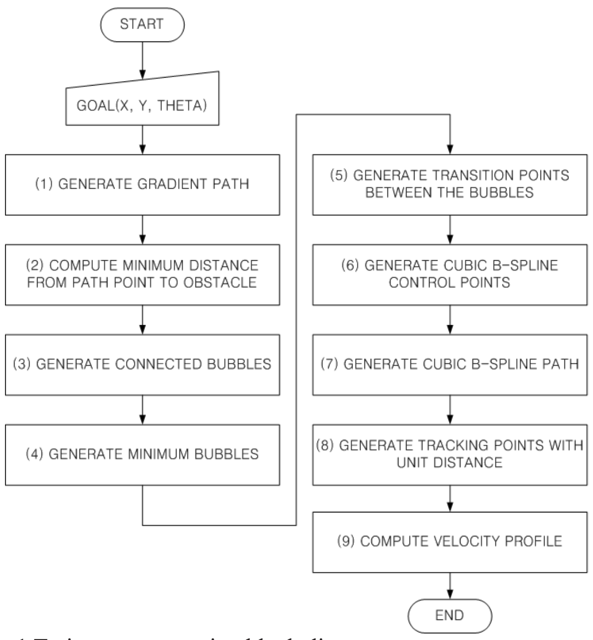
2.2. Contour-tracking behavior

The *contour-tracking* behavior is designed to deal with extremely dynamic situation which relatively large difference with the previous given map and many moving obstacles. Hence, *contour-tracking* is a coordinated wall-following behavior which drives the robot in parallel with the wall, maintaining a constant distance from the wall. The robot's local movement is not affected by localization accuracy. It is sufficient enough to monitor the robot's movement periodically, in order to assume the correct movement toward a goal. *Contour-tracking* is especially useful in environments where topological connectivity is dominant, such as corridors in office buildings.

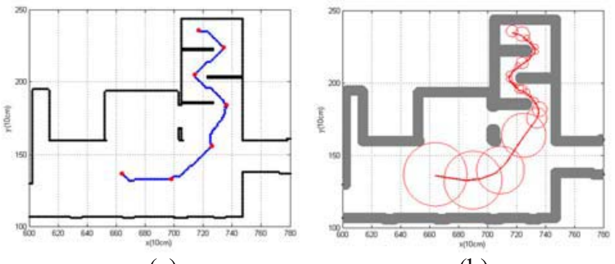
We exploit the Curvature-Velocity method [4] to compute target velocity. However, we are not using the heading and velocity cost function since *contour-tracking* is designed to use in dynamic environments. The heading cost function is not adoptable since the goal heading is computed by the localizer. Also, the velocity cost function is not suitable for dynamic environments.

2.3. Tracking behavior

Our previous research [1] and [7] shows that a mobile robot can improve the performance and reliability of the navigation result using multiple navigation behavior. We use the DWA instead of Trajectory tracking in [7] since it shows more robustness than the Trajectory tracking. In cluttered and complex environments, however, the *AutoMove* have problems such as emergent stop since it sufficiently slow down its speed. Our experiments shows that event the *AutoMove* shows faster than the Trajectory tracking in wide area, but the Trajectory Tracking is faster than *AutoMove* in a narrow and cluttered environment.


START
GOAL(X, Y, THETA)
(1) GENERATE GRADIENT PATH
(2) COMPUTE MINIMUM DISTANCE FROM PATH POINT TO OBSTACLE
(3) GENERATE CONNECTED BUBBLES
(4) GENERATE MINIMUM BUBBLES
(5) GENERATE TRANSITION POINTS BETWEEN THE BUBBLES
(6) GENERATE CUBIC B-SPLINE CONTROL POINTS
(7) GENERATE CUBIC B-SPLINE PATH
(8) GENERATE TRACKING POINTS WITH UNIT DISTANCE
(9) COMPUTE VELOCITY PROFILE
END
Fig. 1 Trajectory generation block diagram

We exploit the Gradient method to generate path. Then using the elastic band [9], we generate continuous curvature path using cubic B-Spline. Fig.1 shows the step to generate the trajectory. At first, we generate a collision-free optimal path using the gradient method. At second step, compute the minimum distance from each point of the generated path. Using the elastic band method, step (3) to (6) generate control point of the gradient path and cubic B-Spline path at step (7). Then we generate trajectory by computing unit distance at step (7). At step (8), we compute the velocity profile which limited by the curvature. Finally, the motion control of the robot is done by Kanayama's tracking controller. [11]



(a) (b)

Fig. 2 An example tracking path (a) a gradient path, (b) cubic B-spline path

3. Navigation framework modeling using the GSPN

3.1. GSPN modeling of the navigation framework

Our strategy to model the system behavior is that each navigation components, localizer, path planner or environmental evaluation components, have each independent sate which modeled by the trap. This modeling strategy is based on modular modeling scheme. Also, each status is connected by transition.

Fig. 3 shows redesigned GSPN model. We add 3 places 9 transitions. The place P9, P10, P11 and the transition t15, t16, t17, t18, t19, t20, t21, t22, t23, t25 are added to model the tracking behavior. We design the GSPN model using the TimeNet [8].

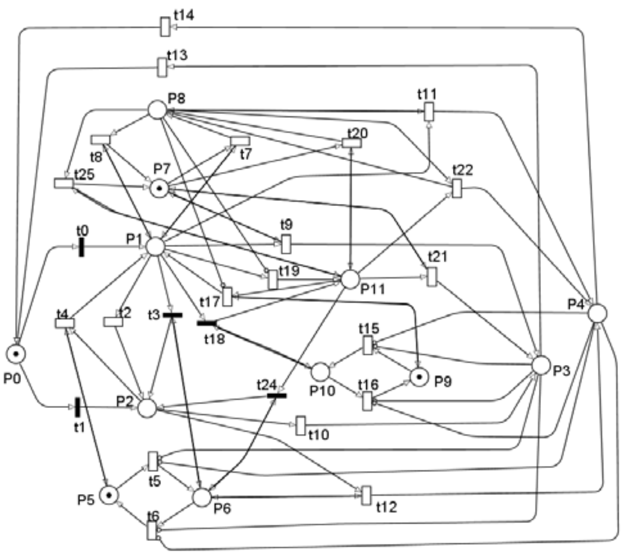


Fig. 3 GSPN Model

The *tracking* behavior is selected when the environmental states are changed into a narrow by t15 fired. When t15 is fired t18 is enabled. Therefore, if the current behavior is *AutoMove*, the behavior immediately changed into *Tracking*. This is done by moving token from P1 to P11. Also, t17 (t19) is enabled when the token is in P11 (P1) and P7. The behavior selection between *AutoMove* and *Tracking* is done by performance analysis.

As pointed out in [1], the modeling process is relatively simple than the direct use of FSA or MPs. The number of full state space including tangible state and vanishing state is 48 states which means if we use the Markov Process Transition Matrix, it requires 48x48 matrixes. The tangible state-space of our previous modeling is just 13 states. Even if just 3 places, 10 transitions and 1 token are increased, the total states are increased into 36 tangible states. This qualitative analysis of the GSPN is useful in real practical modeling because mobile robot navigation architecture requires a lot of components.

Table. 1. Legends for Fig.1

Place	Description
P0	Initial state
P1 (P2)	Running *AutoMove* (*Contour-tracking*)
P3 (P4)	Navigation success (failure)
P5 (P6)	Localizer *Success* (*Warning*)
P7 (P8)	Path planner normal (Abnormal)
P9 (P10)	Navigation path is narrow (wide)
P11	Running Tracking
Transition	**Description**
t0	Start *AutoMove-*
t1	Start *Contour-tracking*
t2 (t4)	Convert to *Contour-tracking* (*AutoMove*) after performance estimation
t3	Convert to Contour-tracking from *AutoMove* due to the localization *Warning*
t5 (t6)	Localization *Warning* (*Success*) event fired
t7 (t8)	Path planner Normal (Abnormal) event fired while the DWA is running
t9 (t10)	*AutoMove* (Contour-tracking) completed
t11	*AutoMove* failed due to no path to the goal
t12	*Contour-tracking* failed due to localization failure
t13, t14	Initialization
t15 (t16)	Path status is changed into narrow (wide)
t17 (t19)	Convert to *AutoMove* (*tracking*) after performance estimation
t18	Convert to tracking due to the path narrow
t20 (t25)	Path planner Normal (Abnormal) event fired while the tracking is running
t21	Tracking completed
t22	Tracking failed due to no path to the goal
t24	Convert to Contour-tracking from tracking due to the localization *Warning*

3.2. Model analysis

In previous section, we focused qualitative the static and dynamic modeling. In this section, we focus on the quantitative analysis of the designed model. The EMC (Embedded Markov Chain) *U* of the GSPN is 36x36 matrixes which generated automatically using the TimeNet [10]. As a result, the total numbers of tangible markings are 36. The tangible marking which enables the transition of t9, t10 and t22 is presented in Table. 2. (M_i =[P0,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11])

Table. 2. Tangible markings for each behavior

Behavior	Markings
AutoMove	M_1=[0,0,1,0,0,1,0,1,0,1,0,0]
Contour-tracking	M13=[0,0,1,0,0,1,0,0,1,1,0,0]
	M14=[0,0,1,0,0,0,1,0,1,1,0,0]
	M15=[0,0,1,0,0,1,0,0,1,0,1,0]
	M16=[0,0,1,0,0,0,1,0,1,0,1,0]
	M30=[0,0,1,0,0,1,0,1,0,0,1,0]
	M31=[0,0,1,0,0,0,1,1,0,0,1,0]

Tracking	M5=[0,0,0,0,0,1,0,1,0,0,1,1]
	M6=[0,0,0,0,0,1,0,1,0,1,0,1]

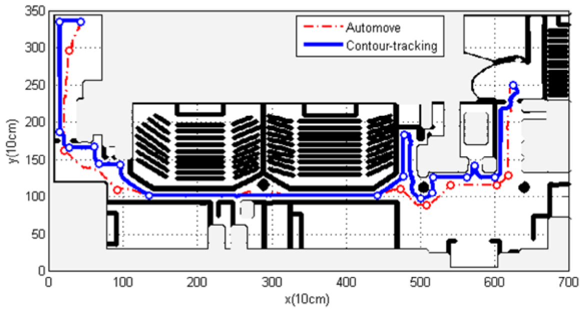
$$Y = YU\sum_{i=1}^{s} y_i = 1, where\, s = 36 \quad (1)$$

$$\pi_i = \frac{y_i m_i}{\sum_{j=1}^{s} y_j m_j},\ m_i = \frac{1}{\sum_{t_k \in E(M_i)} y_j m_j} \quad (2)$$

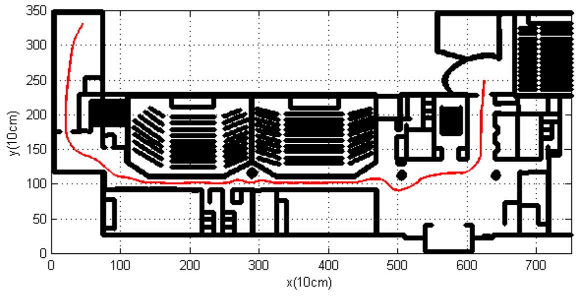
Equation (1) and (2) shows how to compute the limiting probability of the each marking. Equation (1) is linear equation of the EMC matrix. Equation (2) is the limiting probability EMC matrix of each marking.

$$f_{auto} = \pi_1 / t_9$$
$$f_{cont} = (\pi_{13} + \pi_{14} + \pi_{15} + \pi_{16} + \pi_{30} + \pi_{31}) / t_{10} \quad (3)$$
$$f_{track} = (\pi_5 + \pi_6) / t_{22}$$

Basically, the behavior selection is done by two kind of transition. If the immediate transition is enabled, the transition always fired without considering the timed transition. For example, if the t3, t18, t24 is enabled, the behaviors changed into the *Contour-tracking*. The throughput of the each behavior can be computed using the equation (3). Therefore, when the timed transition is enabled with conflict, behavior selection can be done by computing the throughput of each behavior using the equation (3).



(a)



(b)

Fig. 4 (a) *AutoMove* and *Contour-tracking* reference path [7], (b) *Trajectory* tracking reference trajectory.

Fig.4 shows simulated path to compare the quantitative measures. The simulated navigation results are shown in Table. 3. *Tracking* behavior follows optimal tracking path, however, travel speed is lower than the *AutoMove* in most cases.

Table. 3. Simulated travel distance and time

Behavior	Travel Distance(m)	Travel Time(s)
AutoMove	89.4	223.5
Contour-tracking	110.9	369.7
Tracking	92.20	307.32

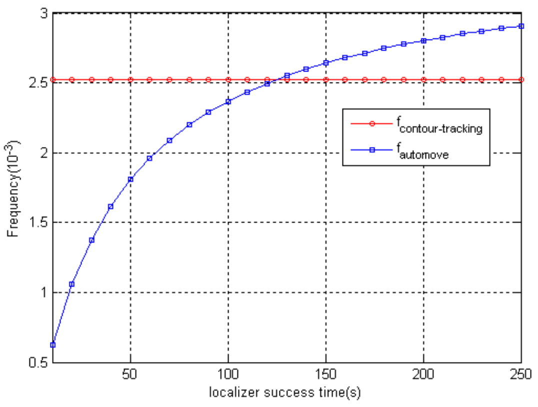


Fig. 5 Simulation result with Localizer success time t9

As shown in fig.5, $f_{automove}$ is proportional with the localizer success time t5. $f_{automove} > f_{contour\text{-}tracking}$ when the localizer success time is larger than 250s which means that the *AutoMove* is efficient than *Contour-tracking*.

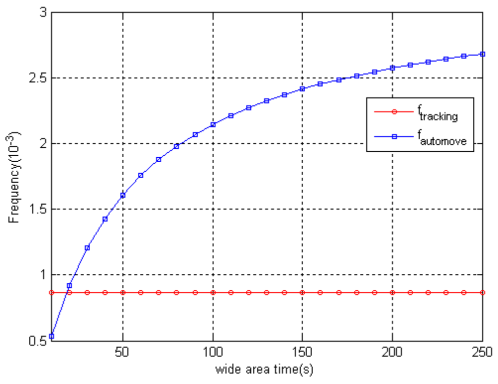


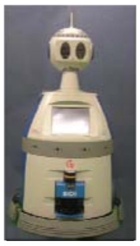
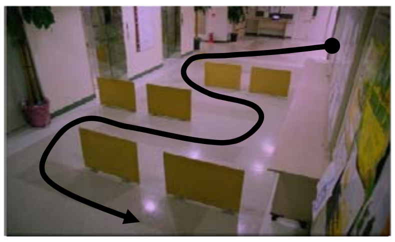
Fig. 6 Simulation result with wide area time t15

As shown in fig.5, $f_{automove}$ is proportional with the wide area time t15. $f_{automove} > f_{tracking}$ when the wide area time is larger than 25s which means that the *AutoMove* is efficient than *Tracking*.

The result of fig.5 and fig. 6 shows that behavior selection can be automatically selected by the performance estimation. This is one of main advantages of the GSPN because as states are increased, it is very hard to manage by the human operator.

4. Experimental Results

4.1. Experimental setup

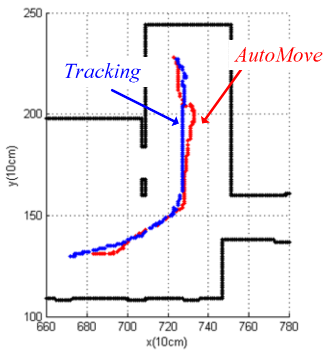



(a) (b)

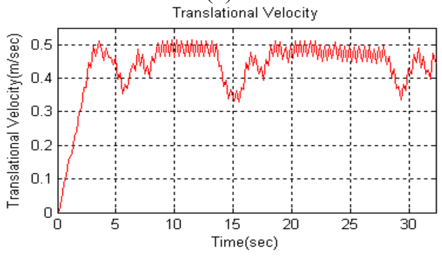
Fig. 7 (a) mobile robot Infotainment, (b) Target workspace to compare the navigation behaviors

Fig.7 (a) shows a robot Infotainment. Experiments were carried out using the robot. Fig. 7 (b) shows target workspace with static obstacles, black-line represents the empty space which roughly shows the tracking path.

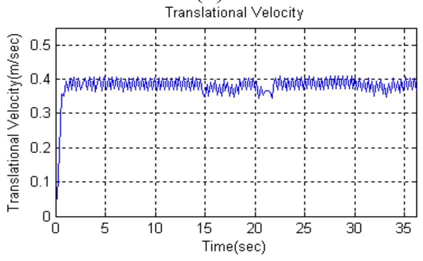
4.2. Navigation comparison between *AutoMove* and *Tracking* behavior



(a)



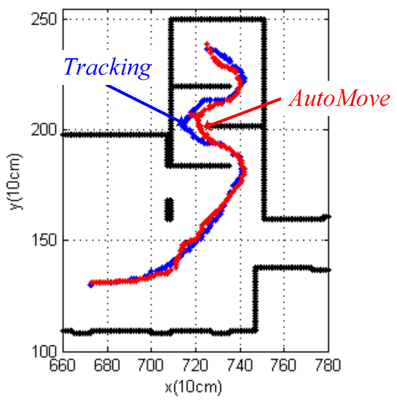
(b)



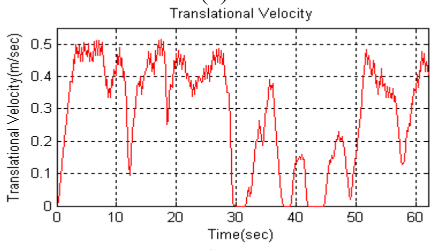
(c)

Fig. 8 Navigation results (a) *AutoMove* and *Tracking* path, (b) *AutoMove* velocity, (c) *Tracking* velocity

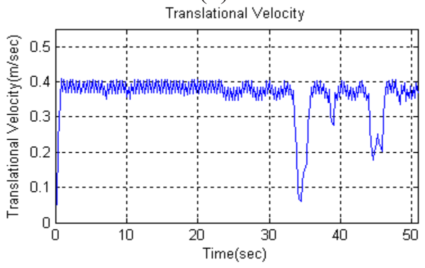
Fig.8 shows wide area navigation experimental result. As shown in fig. 8 (b), (c) the total travel time of *AutoMove* is smaller than Tracking navigation time.



(a)



(b)

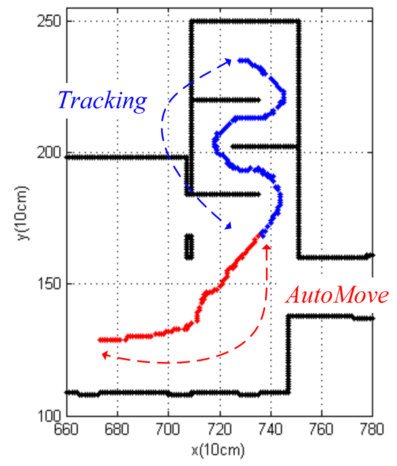


(c)

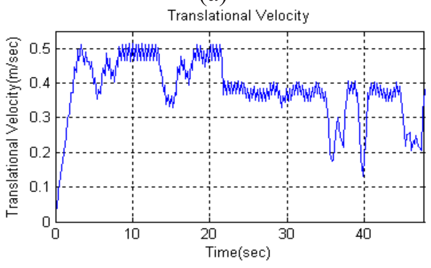
Fig. 9 Navigation results (a) *AutoMove* and *Tracking* path, (b) *AutoMove* velocity, (c) *Tracking* velocity

Fig.9 shows narrow area - requires high rotational velocity - navigation experimental result. The resultant path does not show the significant difference. However, the velocity profile shows that *AutoMove* stops 3 times. As a result, the total *Tracking* travel time is smaller than the *AutoMove* travel time.

4.3. Navigation result using multiple navigation behaviors



(a)



(b)

Fig. 10 Navigation results (a) experimental result path, (b) Velocity profile

Fig.10 shows navigation result using behavior selection. The mobile robot moves from start position using the *AutoMove*. When the t15 fired, the immediate transition t18 is fired. As a result, the navigation behavior changed into the *Tracking*. As shown in the fig. 10 (b), total travel time is decreased into 48s. The experimental result shows that the navigation performance is increased by using the navigation behavior selection.

5. Conclusion

In this paper, we carry out the experimental comparison between the *Tracking* and *AutMove* behavior. The experimental results show that the two navigation schemes have appropriate environments. Therefore, we redesign the navigation framework proposed method in [1]. As pointed out in [1], the main advantage of the GSPN is that the numbers of places and transitions only increase linearly as the system complexity increase, whereas the numbers of states in the MPs increase exponentially. The redesigned model in this paper, EMC (Embedded Markov Chain) matrix has 36x36 dimensions. The system modeling using the GSPN is quite useful, even though the state-space increases exponentially. Also, the performance analysis can be done automatically using mathematical formulation of the GSPN. The experimental results show that the selection between the *Tracking* and *Automove* by newly added components increases the navigation performance.

6. References

[1] G. Kim and Woojin Chung, "Navigation Behavior Selection Using Generalized Stochastic Petri Nets (GSPNs) for a Service Robot," IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, vol. 37, No. 4, July 2007.

[2] J. Wang Timed Petri Nets Theory and Application, Norwell, MA: Kluwer, 1998.

[3] A. Turner, A. Penn, "Making isovists syntactic: isovist integration analysis," International Symposium on Space Syntax, 1999.

[4] R. Simmons, "The Curvature-Velocity Method for Local Obstacle Avoidance," Proc of International Conf. on Robotics and Automation, 1996.

[5] W. Chung, C. Moon, K. Kim and J. Song, "Design of a sensor model and semi-global localization of a mobile service robot," SICE-ICASE International Joint Conference 2006, pp4206-4265, Oct., 2006.

[6] L. Zhang, "Line Segment Based Map Building and Localization Using 2D Laser Range finder," Proc. of International Conf. on Robotics and Automation, vol 3, pp2538, 2000.

[7] C. Moon, W. Chung, "Intelligent navigation behavior selection of a mobile robot," 2007 International Symposium on Humanized Systems.

[8] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," *Proc. of IEEE International Conf. on Robotics and Automation,* 1999.

[9] Sean Quinlan, "Real-time Modification of Collision-free paths," Ph.D Thesis, 1994.

[10] A. Zimmermann, TimeNET 4.0 User Manual, Technische Universität Berlin, 2007, http://pdv.cs.tu-berlin.de/~timenet

[11] Kanayama, Y. Kimura, Y. Miyazaki, F. Noguchi, T., "A stable tracking control method for an autonomous mobile robot," International Conference on Robotics and Automation, 1990.