Integrated Navigation of the Mobile Service Robot in Office Environments

Woojin Chung, Gunhee Kim, Munsang Kim, Chongwon Lee

Advanced Robotics Research Center, Korea Institute of Science and Technology,
39-1 Hawolgok-dong, Sungbuk-ku, Seoul, 136-791, Korea
(Tel : +82-2-958-6743; E-mail: {wjchung, knir38, munsang, cwlee}@kist.re.kr)

Abstract: This paper describes an integrated navigation strategy for the autonomous service robot PSR. The PSR is under development at the KIST for service tasks in indoor public environments. The PSR is a multi-functional mobile-manipulator typed agent, which works in daily life. Major advantages of proposed navigation are as follows: 1) Structured control architecture for a systematic integration of various software modules. A Petri net based configuration design enables stable control flow of a robot. 2) A range sensor based generalized scheme of navigation. Any range sensor can be selectively applied using the proposed navigation scheme. 3) No need for modification of environments. (No use of artificial landmarks.) 4) Hybrid approaches combining reactive behavior as well as deliberative planner, and local grid maps as well as global topological maps. A presented experimental result shows that the proposed navigation scheme is useful for mobile service robot in practical applications.

Keywords: Service robot, Navigation, Localization, Path planning

1. INTRODUCTION

In recent years, there have been various trials to extend robotics technology to non-industrial applications, such as surgery, rehabilitation, floor cleaning, building patrol and so forth. Good examples are a delivery robot Helpmate [1], a building patrol robot Navmaster [2] and a museum tour guide robot Minerva [3]. At the KIST (Korea Institute of Science and Technology), a new mobile robotic system is under development towards indoor public services, as shown in Fig. 1. The PSR is composed of a holonomic omni-directional mobile robot, a six degree-of-freedom PCJ (Passive Compliance Joint) manipulator, a dexterous multi-fingered robot hand and a reconfigurable trailer system. The aim of the project is to develop an intelligent multi-functional servant robot in building environments. A key concept of the mechanism is a reconfigurability based on the modularity, towards various service tasks.

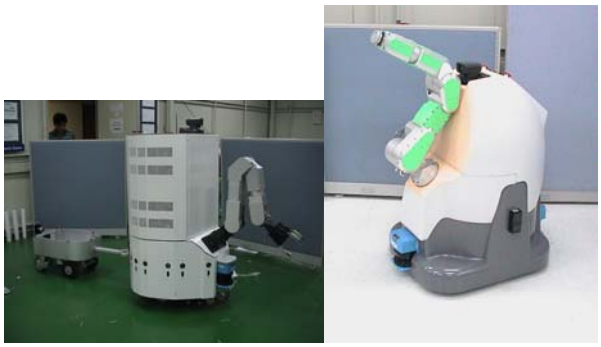

Fig. 1. Prototypes of the indoor public service robot PSR.

Our viewpoint is that the robotic system in daily life is different from the conventional industrial automation problem. Many of the modern buildings are already equipped with automated delivery system such as conveyor and a reliable monitoring system for security purpose. Only a little portion of those applications will be replaced by mobile robotic solutions. Therefore, a mobile robotic agent should be a multi-functional servant, who can maximally utilize its capability towards various applications. Our major target is to achieve a delivery, a patrol, a guide and a floor cleaning by one robotic system. A mobile manipulator system can be a useful solution for various tasks, while specific automation equipment cannot be applied to other application. So far, a trailer system design and control [4], a control architecture design [5] and a localization strategy of a mobile robot [6] were already reported.

In this paper, a major scope is to explain a strategy on the navigation synthesis of a mobile robot. After the brief description of the PSR (KIST Public Service Robot) platform in Chapter 2, integrated navigation strategy will be explained in Chapter 3. Chapter 3 illustrates an overview of the developed navigation system first, and then explains key navigation modules, maps, the localizer, the path planners, and the tracking behavior. Chapter 3 also introduces control architecture and a system integration methodology. The results of experiments are discussed in Section 4. Finally some concluding remarks are given in Chapter 5.

2. HARDWARE STRUCTURE OF MOBILE ROBOT

There are several types of sensors in the PSR for autonomous navigation. In order to cover all around the PSR, two laser range scanners are set up on front and rear sides, and two infrared scanners are equipped on left and right sides. A gyroscope sensor additionally gives accurate orientation information for better localization performance.

Holonomic and omni-directional movements are essential requirements in the PSR system. It provides control simplicity by excluding nonholonomic mechanical constraints. Therefore, it is possible to allow rich behaviors to the PSR in various applications, such as autonomous navigation, mobile manipulation and trailer control. The double-offset active orientable wheel mechanism with conventional tires is proposed for the PSR as shown in Fig. 2. It elevates reliability and durability since it has simple mechanical structure. Moreover, using conventional tires leads accurate positioning performances regardless of floor conditions.

The proposed wheel mechanism employs two motors, one for steering and the other for driving. It has both longitudinal

and lateral offsets with respect to the pivot point, *i.e.* steering center of the wheel module. Appropriate longitudinal and lateral offsets are set for well-conditioned kinematic model, which means the kinematic mapping between the inputs of motors and the resultant velocities applied to the mobile base is well-conditioned. Suitable design of kinematic parameter enables driving and steering velocity components at pivot point to be orthogonal. Thus, the double-offset wheel mechanism can be driven faster than the other mechanisms with the limited maximum velocity of the motor. The detailed explanation of design requirements, kinematic analysis, a fabricated prototype, and preliminary experiments are reported in [7].

Fig.2. prototype of double-offset active orientable wheel mechanism

3. INTEGRATED NAVIGATION SYSTEM

3.1 Overview of integrated navigation system

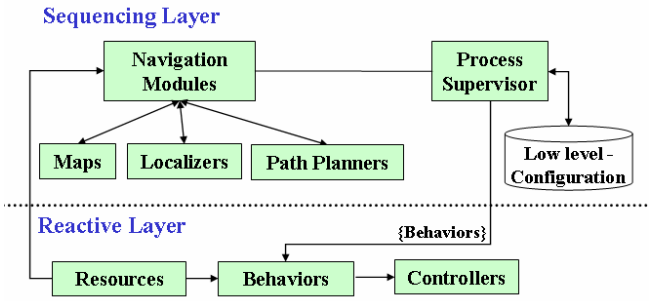

Fig.3. Overview of Integrated navigation system

Fig. 3 shows some key components of our integrated navigation system, which is classified into three parts, which are an information part, a reactive part, and a low-level planning part.

The *Navigation modules* are the information part that extracts perceptually meaningful information from raw sensor data using a various kinds of sophisticated time-consuming algorithms. The main components are *maps*, *localizers*, and *path planners*.

The reactive layer controls the robot motion by executing relatively simple computation in real-time. The *resources* provide framework for efficient fusion of different kinds of sensor data. The *behavior*, a basic motion generator, tightly coupled with sensors and actuators. The *controller* takes commands from the *behavior* and drives the actuators.

The low-level planning part consists of the *process supervisor* and the *low-level configuration*. The *process supervisor* executes internal processes transmitted from the central planner by referring to *low-level configuration* that encapsulates a set of behaviors, related parameters, and error recovery logics. The internal process is defined as a job that is performed by configuring reactive components. In our architecture, the task given by a user is carried out by continuation of internal processes, each of which is composed of several behaviors. The Petri nets based approach is adopted for the design of the *low-level configuration* [5]. Petri nets are useful tools for the design of configurations since it can clearly represent the relationship between the phase of robot behaviors and the events from the *navigation modules*. The designed Petri net model of our fundamental navigation process *PrAutoMove* will be exampled in the section *F*.

To explain the mechanism of our navigation system, a simple transportation task is exampled. If a robot receives a task, which is transporting the target object to a far-off room, the global path planner initially generates a node path set based on a topological map. Each node point becomes the goal position of a navigation process *PrAutoMove*. By iterating these processes, the robot can reach desired position regardless of how far it is located. If the robot senses the planned path is obstructed in the middle of navigation, then the topological map is updated and the global path planning is performed again.

The PSR is an integrated multi-functional robotic system. Therefore, there are some significant issues of architecture design. It includes information connectivity between a variety of modules, scheduling of information processing, combination of reactivity and deliberation. The detailed description of whole control architecture of PSR is introduced in [5].

3.2 Maps

The map, which is a basic component of navigation, is important both to a user and to a robot. Through the map, a user can obtain the outlined environmental information which is necessary to command his or her intentions. A robot completes a navigation task by performing localization and path planning based on maps. To meet these functional requirements, our navigation system possesses four different types of maps, which are information maps, topological maps, local grid maps, and active maps.

The information map displays user-friendly environmental information to be used in the user interface display. For example, a user can discover where the room A or a rest room is located through the information map.

The topological map compactly describes the entire environments with nodes and arcs. It provides physical location of nodes and the connectivity between them for global path planning.

The local grid map contains the accurate information of the surroundings of a robot. It is used for time-consuming localization algorithm, which will be described in next subsection. The local grid map represents environments by evenly-spaced grid cells. The resolution of our local grid map is 10 cm. The grid value is assigned to each cell of the local grid map for representing the characteristics of environments such as the presence of obstacles, the state of a floor, and a protruding wall. The value of each grid also reflects types of detectable sensors. For example, the cell sensed by the laser scanner has different value with that detected by ultrasonic sensors.

The whole workspace is represented by several patches of local grid maps in our system. The reason for the separation is to reduce the computational burden. Since the implemented localization algorithm is performed by map-matching technique, computing time is closely connected to a map size.

Therefore, the PSR loads an appropriate local map for localization according to where it is, that is, which node is nearest one. By using both a topological and a grid map in the proposed way, advantages of both approaches can be fully exploited.

The local map can be generated by using floor plans directly. Also, it is possible to use the range image registration technique, one of semi-autonomous map building algorithm, developed by us. The range image registration technique is quite a convenient method since the local map is constructed by driving the robot through the environment manually. However, it may potentially fail to describe the environment thoroughly. For example, glass or descending stairs can be omitted since the technique is mainly based on 2-D laser scan data. Once the local map is constructed, it is not changed during execution of the navigation process.

The active map is defined for local path planning. It has similar structure with the local grid map since it is created by loading a part of the grid local map. After the loading, the active map is updated in run-time by using different kinds of sensor data independently. The Histogramic In-Motion Mapping (HIMM) [8] is applied for real-time map updating. The map updating rate depends on the sampling time of each sensor. For example, the active map is updated by laser scanners every 200 msec. The active map is created when a navigation process *PrAutoMove* is started, and destroyed when it terminates. Thus, the active map is set up at starting node point, which becomes the center point of the active map with a 10 m × 10m size. Since the local navigation process last until the robot reaches the target node, two adjacent node points should be located on a single active map.

3.3 Localizer

Our localization method is a map-matching scheme using scanned range data, without using any artificial landmark. A probabilistic position estimation scheme is designed based on Monte Carlo localization (MCL). Two measure functions, Range Image Similarity Measure Function (RISF) and Angular Similarity Measure Function (ASF), are developed for computing positional probabilities. The robot automatically decides whether it uses geometric pattern matching (i.e. walls, pillars) by Hough transform. The details of the reliable position estimation method of the PSR are introduced in [6]. The advantages of the proposed localization scheme as follows.

1) The localization algorithm does not fail even if the sensor readings are partially corrupted.

2) Use of artificial landmarks is not required. In other words, the environment is not changed for the service robot.

3) This algorithm provides solutions for both local tracking problems and global position estimation problems through MCL.

4) The scheme is applicable in both polygonal and non-polygonal environments. ASF provides fast sample convergence and high accuracy in polygonal environments. RISF makes position estimation in non-polygonal surroundings possible.

5) Our method can deal with uncertainties like inaccurate dead reckoning information and sensor noises.

6) The algorithm is a generalized method based on range sensors. Thus, any range sensor can be selectively applied.

3.4 Path planner

The implemented path planner modules focus on the reliable navigation in dynamic environments. Therefore, the path planners are developed by considering various situations that frequently occur in real applications.

The global path planning generates optimum node path set in large-scale environment. It is designed by applying wavefront algorithm [9]. We mainly explain the local path planner from now, because the global path planner is a simple extension of a local one.

The local path planner generates the shortest collision-free path between node points by using both Latombe's wavefront algorithm [9] and Konolige's gradient method [10]. The applied algorithms assume that the environment is static and completely known. These assumptions are feasible in our navigation system, since the update of the active map is frequent enough. The planned path is represented by an ordered set of points on the active map. The points are contiguous and do not have to be located on sampled grid.

Fig. 4 shows our local path planning algorithms. The local path planner repeats this algorithm until the local navigation process is completed.

Step 1: Gathering all necessary input data
(1) Obtaining the current position of a robot
(2) Setting target node point as the goal set.
(3) Identifying obstacle set from the active map.

Step 2: Building navigation function.
(1) Calculating intrinsic cost.
(2) Calculating adjacency cost.

Step 3: Checking error states
(1) Checking whether the original goal is restored.
(only if the original goal has been changed.)
(2) Checking whether the goal is occupied.
(3) Checking whether no path to the goal exists.

Step 4: Generating path set (only if one or more of path updating rules are satisfied.)
(1) Constructing path set by steepest descent method.
(2) Smoothing the path using Bezier interpolation.
(3) Performing velocity planning.

Fig.4. Local path planning algorithm

The first step is to gather all input data required to execute the local path planning. They include the absolute current position of a robot, a goal set, and an obstacle set. In most cases, the goal set has one element, which is the coordinates of target node point of the local navigation process. The obstacle set is obtained by interpreting the active map. It contains the coordinates of obstacle points that are enlarged by robot radius since the path planning algorithm assumes a robot to be a point.

The second step is the generation of the navigation function. In the gradient method, the path cost is defined as the sum of an intrinsic cost and an adjacency cost. The intrinsic cost means the cost of traversing through the given point. It represents the characteristics of a corresponding point such as obstacle regions or slippery regions. The intrinsic cost is extremely large at an obstacle point to prevent the collision with an obstacle. Moreover, the path behavior around an obstacle can be controlled by the assignment of intrinsic cost. The adjacency cost implies the path length. Thus, it increases proportionally as the distance with the goal position grows. The navigation function is assigned so that each point in a configuration space, which corresponds to the active map in

our system, has the minimal path cost. Then, the optimal path is computed by moving in the direction of the gradient of the navigation function. The detailed explanation of the navigation function is given in [9] and [10].

The third step is to check whether the path planner falls into error states. Our algorithm concerns about two errors. One is path blocking, and the other is the occupation of a goal position by obstacles.

The Path blocking implies that there is no path between two adjacent nodes. If the path blocking arises, the robot stops moving and waits until new collision-free path is computed. If the path is still obstructed after a pre-determined time, the local path planner gives up the navigation process. It is reported to the central planner and causes the update of the topological map and regeneration of the path node set by the global path planner. The occurrence of this error can be automatically detected by checking the navigation function.

The occurrence of the goal occupation is recognized if the intrinsic cost of the goal position is greater than zero. The goal occupation by unknown obstacles is resolved by setting a temporary intermediate goal. Then, the path planner checks continuously removal of the obstacles on original goal during the motion. When the robot reaches the temporary goal and stays at this point for a long time, it gives up reaching the original goal. Then, it decides whether it is necessary to set another intermediate goal between next node and current position. By iteration of these actions, the robot can successfully navigate between nodes.

The last step is to generate path set based on the navigation function. However, the path set is not updated every time since it is not desirable to change the path set too frequently. It can cause the instability of path tracking control. Therefore, the rough path set is initially computed by considering only grid points in active map. Then, detailed generating process is executed only if one or more of following path updating rules is satisfied.

1) First computation of path set: It means that any path set has not been constructed for the current navigation process.

2) The change of the goal position: This occasion arises if a temporary goal is set by an obstacle occupation or the original goal is restored by removal of an obstacle.

3) The detection of an obstacle on some points of the existing path set: If the existing path is obstructed partially by a new sensed obstacle, the path set is updated.

4) The remarkable decrease of the number of elements in an existing path set: It implies that the path planner discovers a new path shorter than the old one.

The optimal path is obtained by applying steepest descent method to the navigation function. The smoothing process follows so as to improve tracking performance. The Bezier interpolation is adopted in our algorithm. It makes it possible to smooth the path without loss of geometric continuity in a relatively simple manner. The resultant path set also contains the information about robot velocities. The distance between two adjacent points in the path set is proportional to the magnitude of the velocity.

On a main computer of PSR, a 800 MHz Pentium processor PC, it takes about 0.6 second on average to execute the whole path planning algorithm shown in Fig. 4. The generation of the navigation function, step 2 of the algorithm, dominates the computation time. The times for step 1 and sep2 is negligible, less than 1 msec., and step 3 requires about 100 msec. Note that they are measured while the PSR performs an actual service task. It means that this result is obtained while the PSR activates most components including several time-consuming algorithms such as localization and reactive control loops. We can conclude that the computing time of implemented path planning algorithm is acceptable for applying the mobile robot systems that serve in dynamic environments.

3.5 Behavior *BhAutoMove*

The behavior *BhAutoMove* generates a robot motion by tracking the path set from the local path planner. It also reports the results of a local navigation process.

The behavior *BhAutoMove* tracks the elements of the path set one by one every sampling loop. Since our mobile platform is set free from nonholonomic constraints, the simple PID control is adopted for tracking control.

In real applications, the discontinuity of tracking control occurs frequently. It mainly results from position updates by the localizer. Therefore, the behavior also contains several schemes for stable tracking such as an acceleration filter.

The path planner specifies only two-dimensional motion. Thus, the behavior can freely decide a robot's rotational movement. When the robot pulls trailers, the angular velocity can be assigned so as to behave like a conventional two wheel robot. In other case, the robot can follow the wall with facing to it for stable localization.

3.6 Configuration design for process *PrAutoMove*

Fig. 5 and Table 2 represent the Petri net model of the process *PrAutomove* in the *low-level configuration*. The process *PrAutomove* consists of one essential behavior *BhAutomove*, and several error states and fault states. The error states defined in the local path planner and the localizer are also considered. If the robot falls into an error state, the recovery process is executed. However, if the problem is unresolved in spite of its effort, that is, the token is assigned to a fault place in Petri net model, the *process supervisor* announces the failure of a given process and terminates the execution. This result is reported to the *planner*, which reconfigures the planned sequence of processes to accomplish the task commanded from a user.

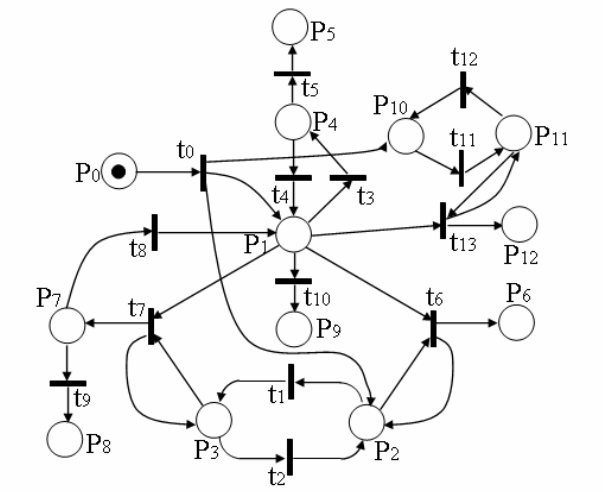

Fig.5. Petri net model of process PrAutomove

Table 1 Description of places and transition of Fig. 5

	Description
P_0	Standby
P_1	Executing the behavior *BhAutomove*
P_2	State: The original goal has NOT been changed
P_3	State: The original goal has been changed
P_4	Error: The path to the goal does not exist. (The robot stops moving and waits until new path is detected)
P_5	Fault: The path to the goal does not exist
P_6	Completing of the process *PrAutoMove*
P_7	Error: Reach the temporary goal and stop moving until the original goal is restored
P_8	Fault: Reaching the original goal is failed since it is occupied by obstacles
P_9	Error: The behavior *BhAutomove* CANNOT extract the desired point from path set (Reach the last point in the path set)
P_{10}	State: Normal localization (Localizer updates a robot position periodically.)
P_{11}	State: Abnormal localization (Localizer doesn't updates a robot position, and robot navigates only by using odometry)
P_{12}	Fault: A robot fails to find out its own position.
t_0	*Process supervisor*(*PS*) starts the behavior *BhAutomove*
t_1	*LPP* changes the original goal due to the goal occupation
t_2	*LPP* restores goal position since the obstacle on the original goal is removed
t_3	*LPP* detects the error "no path to the goal"
t_4	*LPP* finds out the path to the goal
t_5	*PS* terminates the behavior *BhAutomove* due to "No path to goal exists"
t_6	*PS* completes the Behavior *BhAutomove*
t_7	*PS* reaches the changed goal and stop moving.
t_8	*LPP* detects a new goal which is the original goal or another temporary goal.
t_9	*PS* terminates the behavior *BhAutomove.* (The Robot fails to reach the original goal)
t_{10}	The behavior *BhAutomove* CANNOT extract the desired point from path set
t_{11}	Localizer finds out the estimated position is accord with the actual robot position.
t_{12}	Localizer finds out the estimated position is not accord with the actual robot position.
t_{13}	*PS* terminates the behavior *BhAutomove* due to the failure of *localization*

Petri nets are advantageous for controlling a navigation job, because the activities on the configuration level are event-driven and the state of a system can be divided into the possible discrete states. As a graphical tool, Petri nets can visualize the control logics in configurations and make it easy to check the states of systems. Moreover, the dynamic change of states can be explicitly described since the Petri nets exploit tokens, which clearly indicate state transition with respect to event firing. As a mathematical tool, it is possible to set up mathematical models and analyze the designed logics. Therefore, it is convenient to define an efficient error handling schemes.

4. EXPERIMENTS AND RESULTS

The developed navigation system is implemented and tested on the PSR-II platform. The scenario is a navigation task in conventional office room. The size of the room is about 6 m × 7 m. The local grid map of the workspace and is shown in Fig. 6. The PSR starts at node 0 and return to the initial position through node1 and node 2. Therefore, the task consists of three local navigation processes.

We represent the results of an experiment by showing a navigation process from node 1 to node 2. The actual trajectory is also represented in Fig. 6. The path planning and localization is explained in Fig. 7 and Fig. 8, respectively.

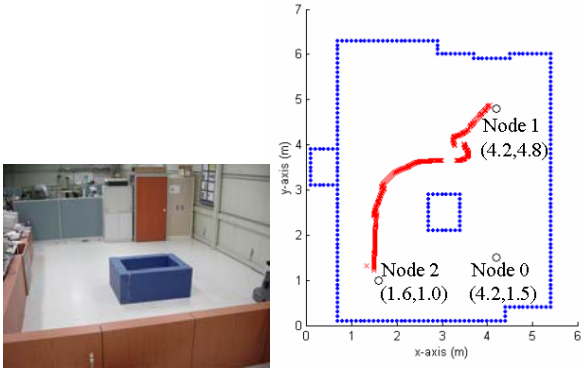

Fig.6. Experimental environments

Fig. 7 shows some active maps and the planned paths during the experiment. The initial position of the planned path is the current position of the PSR, and the end point is a goal position. A user stands intentionally in the PSR's way several times. The collision-free optimal path is changed according to the locations of obstacles. If the user walk in the middle of the passageway as represented in Fig. 7.(c), the PSR takes a roundabout way. Note that the obstacle points are enlarged by robot radius and intrinsic cost region. Thus, a robot moves with keeping a certain distance.

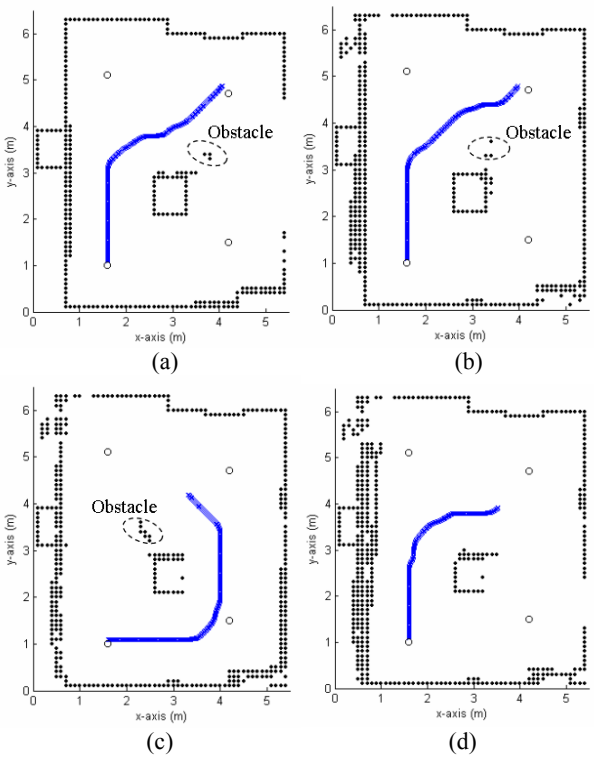

Fig.7. Path planning process during the navigation from node 1 to node 2

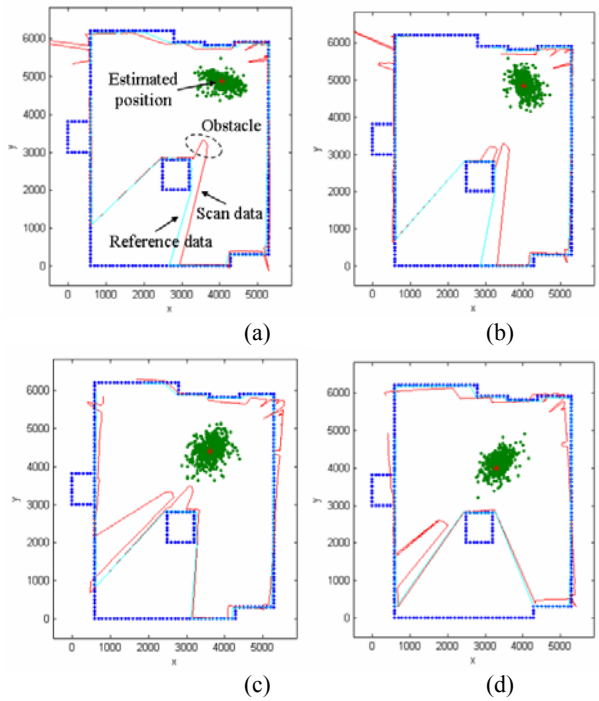

Fig.8. Localization process during the navigation from node 1 to node 2

Fig. 8 shows the local map, laser scan data, reference data, sample distributions, and estimated position each calculation. The data for Fig. 7 and Fig. 8 is gathered simultaneously with a single experiment. As The reference measurements of the estimated robot location are mostly consistent with the scan measurements of actual robot position. It means that the accurate position estimation is accomplished. Although a user disturbs the PSR's navigation, the proposed localization algorithms work successfully. Fig. 8 shows that sensor measurements are partially corrupted due to dynamic objects.

5. CONCLUSION

This paper presented an integrated navigation strategy of the service robot PSR. Major advantages of the proposed system can be summarized as follows:

1) Structured control architecture for a systematic integration of various software modules: Appropriate design of the Petri net based configuration surely guarantees reliable robot's operation.

2) A range sensor based generalized scheme of navigation: Range sensors are implemented for obstacle detection, path planning and localization. A multiple role of a map was successfully implemented for environmental representation, a reference database of a localization, as well as optimal path generation to the goal. There is no fundamental limitations of using any of ultrasonic, infrared or laser range sensor, because the proposed scheme is applicable for any type of range sensors.

3) No need for modification of the environment: Navigation can be carried out without putting any artificial landmark.

4) Hybrid approaches: The navigation includes reactive behavior as well as deliberative planner, and local grid maps as well as global topological maps.

The PSR is currently under the field test. A delivery task was successfully carried out by autonomous navigation in an office building, where operation space is 80m X 35m. The environment is composed of corridors and office rooms.

ACKNOWLEDGMENTS

This research is supported by the grant of the "Development of a service robot technology" project, Ministry of Science and Technology, Korea. The authors acknowledge Dr. Kurt Konolige at the SRI for the fruitful discussions, and Hyundai Heavy Industry Company for prototyping the PSR.

REFERENCES

[1] Gay Engelberger, "Helpmate, a service robot with experience", Industrial Robot, V25, No2. pp 101-104. 1998.
[2] http://www.cybermotion.com/.
[3] S. Thrun, et al. Proc. of the IEEE International Conference on Robotics and Automation (ICRA'99), 1999.
[4] Jaehyung Lee, Woojin Chung, Munsang Kim, Chongwon Lee, and Jeabok Song, "A passive multiple trailer system for indoor service robots." In Proc. of IROS 2001, pages 827--832, Maui, Hawaii, Oct 2001.
[5] Gunhee Kim, Woojin Chung, Munsang Kim, Chongwon Lee, "Tripodal Schematic Design of the Control Architecture for the Service Robot PSR," in Proceeding of the IEEE Conference on Robotics and Automation, Taipei, Taiwan, 2002.
[6] Dongheui Lee, Woojin Chung, Munsang Kim, "A Reliable Position Estimation Method of the Service Robot by Map Matching," in Proceeding of the IEEE Conference on Robotics and Automation, Taipei, Taiwan, 2002.
[7] Woojin Chung, Kihwan Kim, Jaehyung Lee, Taigun Lee, Bongsoo Kang, Munsang Kim, and Chonwon Lee, "Design and Control of the Indoor Public Service Robot," in Proceeding of the IFAC Workshop on Mobile Robot Technology, Jejudo, Korea, pp.303-308, 2001.
[8] J. Borenstein and Y. Koren, "Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance," IEEE Trans. on Robotics and Automation, vol. 7, no. 4, Aug. 1991, pp. 535-539.
[9] Jean Claude Latombe, "Robot Motion Planning," Kluwer Academic Publishers, MA; 1991, pp.295.
[10] Kurt Konolige, "A Gradient Method for Realtime Robot Control," in Proceeding of the IEEE/RSJ Conference on Intelligent Robots and Systems, Takamatsu, Japan, pp.639-646 2000.