An automatic switching approach of robotic components for improving robot localization reliability in complicated environment

Wenshan Wang, Qixin Cao and Xiaoxiao Zhu
Research Institute of Robotics, Shanghai Jiao Tong University, Shanghai, China, and

Masaru Adachi
Yaskawa Electric Corporation, Fukuoka, Japan

Abstract

Purpose – Robot localization technology has been widely studied for decades and a lot of remarkable approaches have been developed. However, in practice, this technology has hardly been applied to common day-to-day deployment scenarios. The purpose of this paper is to present a novel approach that focuses on improving the localization robustness in complicated environment.

Design/methodology/approach – The localization robustness is improved by dynamically switching the localization components (such as the environmental camera, the laser range finder and the depth camera). As the components are highly heterogeneous, they are developed under the robotic technology component (RTC) framework. This simplifies the developing process by increasing the potential for reusability and future expansion. To realize this switching, the localization reliability for each component is modeled, and a configuration method for dynamically selecting dependable components at run-time is presented.

Findings – The experimental results show that this approach significantly decreases robot lost situation in the complicated environment. The robustness is further enhanced through the cooperation of heterogeneous localization components.

Originality/value – A multi-component automatic switching approach for robot localization system is developed and described in this paper. The reliability of this system is proved to be a substantial improvement over single-component localization techniques.

Keywords Middleware, Localization, Mobile robot, Multi-component

Paper type Research paper

1. Introduction

Localization is one of the most fundamental and important functions required by the autonomous mobile robot. As a result, this subject has been widely studied for decades and many different approaches have been developed. The essence of the localization problem is to estimate the robot's position and orientation based on the data available from sensors that are located either on the robot or somewhere in the environment. Frequently used sensors are odometers, cameras (Davison *et al.*, 2007), microphone arrays (Huang *et al.*, 1999), lasers (Skrzypczynski, 2012; Teslić *et al.*, 2010), depth cameras (Biswas and Veloso, 2012), etc. The estimation process is usually based on a probabilistic algorithm such as extended Kalman filter (EKF) (Chenavier and Crowley, 1992; Delalloche *et al.*, 2001), Markov localization (Fox *et al.*, 1998), multiple hypothesis tracking (MHT) (Jensfelt and Kristensen, 2001) and particle filter (Thrun *et al.*, 2001).

Finding a robust and efficient algorithm is always a difficult problem in localization research. This is due to the limitations of the robot capability; the disturbance of the environment uncertainty; as well as the explosion of computation complexity when improving precision in large-scale problem. Even though a lot of research has been conducted in this field, each of the resulting localization methods is usually confined to a particular environment or a specific application. For example, the method based on the use of a camera requires good lighting conditions (Drocourt *et al.*, 1999; Rekleitis *et al.*, 2006); another approach depending on the laser sensor may fail in feature-less environment such as a long corridor (Thrun *et al.*, 2000); and other methods based on the depth camera (Biswas and Veloso, 2012; Endres *et al.*, 2012) cannot deal with the out-door tasks. Thus, for purposes of applying the localization technique to a complicated environment such as day-to-day life scenarios, this paper propose an automatic switching approach among several heterogeneous sensors coupled with localization methods to improve the localization robustness.

For robot localization, it is very common to find several sensors and techniques combined together to achieve better quality. One of the most widely studied methods is one based on the combination of a laser range finder (LRF) and an odometer (Fox *et al.*, 2001). Besides that, an integrated localization system for mobile robots in underground environments, based on a strap-down inertial measurement unit and a digital

The current issue and full text archive of this journal is available at **www.emeraldinsight.com/0143-991X.htm**

Industrial Robot: An International Journal
41/2 (2014) 135–144

[DOI 10.1108/IR-04-2013-338]

This research has been supported by Yaskawa Electric Corporation. It has also been supported by the national major project of advanced manufacturing technical field (2012AA041403).

compass has also been developed for exploration and rescue missions in coal mines and tunnels (Xiong *et al.*, 2009). A multi-sensor system combining data from infrared and ultrasonic sensors for robot navigation has also been implemented for coal mine rescue (Meng *et al.*, 2011). There has also been some research on the data fusion of multiple sensors mostly by means of probabilistic methods (Chen *et al.*, 2008).

However, few studies have considered utilizing modularized methods to develop and manager various localization components. Such components may be highly heterogeneous both with regard to hardware platform and software implementation. Take our system for example, the approach based on the environmental camera is developed using Java in Windows and implemented on an off-board computer whereas the one based on the LRF sensor is realized with C++ in Linux on an on-board computer, etc. There is little information available in literature on how to combine these heterogeneous localization methods for the purpose of improving the localization robustness and reliability in a complicated environment. Thus, this paper present a modularized framework that attempts to improve the system extendibility while also to reduce development complexity.

Considering the foregoing, a middleware technology is introduced into the localization domain. It encapsulates heterogeneous localization components into uniform standardized components. Much research effort has been expended on the middleware technology for heterogeneous robotics system (Mohamed *et al.*, 2009). Player framework provides infrastructure, drivers and basic algorithms for mobile robotic tasks (Collett *et al.*, 2005); lime is a more specialized library that is slanted more towards sensor networks (Murphy *et al.*, 2006); miro is an object-oriented middleware for mobile robots based on the common object request broker architecture (CORBA) (Utz *et al.*, 2002). Another middleware example based on CORBA is RT middleware (Ando *et al.*, 2005, 2008), which provides the necessary services to enable implementing robotic applications in distributed systems. We argue that the middleware technology needs to offer the full development lifecycle for robot software. That is, it needs to have the capacity to provide not only the communication model but also the interfaces for managing component states. With this in mind, a distributed localization system using robot technology component (RTC) is proposed.

The approach should be useful in the event that each localization technique is individually unable to accomplish the localization task. For example, our LRF-and-odometer-based localization method always fails in the corridor. Therefore, an extra environmental camera is installed on the ceiling of the corridor. By means of the automatic switching approach, the localization lost situation is significantly decreased. Another advantage of this framework is that it decouples different localization methods. Consequently, this allows for relatively hassle-free incorporation of new methods while still maintaining the old.

The paper is organized as follows. In Section 2, the robot and localization components used in the experiment are presented. Section 3 the modular developing framework based on RTC is discussed. The problem of improving the localization robustness by dynamically changing the connection of the components is described in Section 4. The results of the experiments are presented in Section 5. Also in Section 5 some comparisons are given to show the efficiency of our method. Finally, conclusions are drawn in Section 6.

2. System architecture and localization approaches

In this paper, the study is based on a distributed robotic platform (Figure 1). One of the main advantages is that it is convenient to expand the system with new localization modules without any undue concern of implementation language or environment. Also, the person responsible for one module can be assured that the modification of his module will not affect the others. This study makes use of a mobile robot equipped with a "Kinect" depth camera and a "Hokuyo" LRF sensor. Besides these, three environmental cameras are mounted on the ceiling. One is located in the corridor, one in the indoor room, and another in the outdoor square (Figure 2). To combine different approaches to form a robust localization system, each module is designed to report its localization performance based on its own evaluation. This parameter – which reflects each module's confidence in its own localization accuracy – is herein referred to as the localization reliability. For the reason that different localization methods are implemented with different algorithms, it is not likely that this confidence will be evaluated in an identical way for each method. The various detailed models are described below. Effort has been made to make the reliability consistent with the localization accuracy. The localization approach of each component is briefly presented below with a particular focus on the modeling and calculation of the reliability for each localization component.

2.1 Environmental camera based approach

The three web cameras are installed in different locations, in order to effectively provide the positioning information. Since this study does not focus on the concrete localization method, but rather exploration of a method of improving the robustness through the coordination of a variety of localization modules, we employ a simple color tracking approach and assume that there is no color interference during the experiments.

The camera is obliquely installed on the ceiling as Figure 3 shown. This means the accuracy decreases as the robot moves far from the camera. The decrease is mainly due to two reasons. One is the fact that perspective makes the distant robot harder to recognize. The other is the recognition error in the algorithm. For the convenience of deducing the reliability model, the camera is well calibrated, in order to minimize the image distortion.

As Figure 3 shows, let the installation angle of the camera be α, the height be h, the vertical field angle be θ, and the pixel number in vertical direction be N. The field angle for one pixel is given by:

$$\theta_0 = \frac{\theta}{N} \tag{1}$$

P denotes a point on the image, and the vertical location is represented as the angular coordinates $\varphi \in [-\theta/2, \theta/2]$.

The actual distance from camera is:

$$x = \tan(\varphi + \alpha) \cdot h \tag{2}$$

The actual length corresponding to one pixel at the point P is:

$$\begin{aligned} D(\varphi) &= x_\Delta - x = \tan(\varphi + \alpha + \theta_0) \cdot h - \tan(\varphi + \alpha) \cdot h \\ &= \frac{\sec^2(\varphi + \alpha)}{\cot\theta_0 - \tan(\varphi + \alpha)} \cdot h \end{aligned} \tag{3}$$

Figure 1 The architecture of the localization system based on modular development

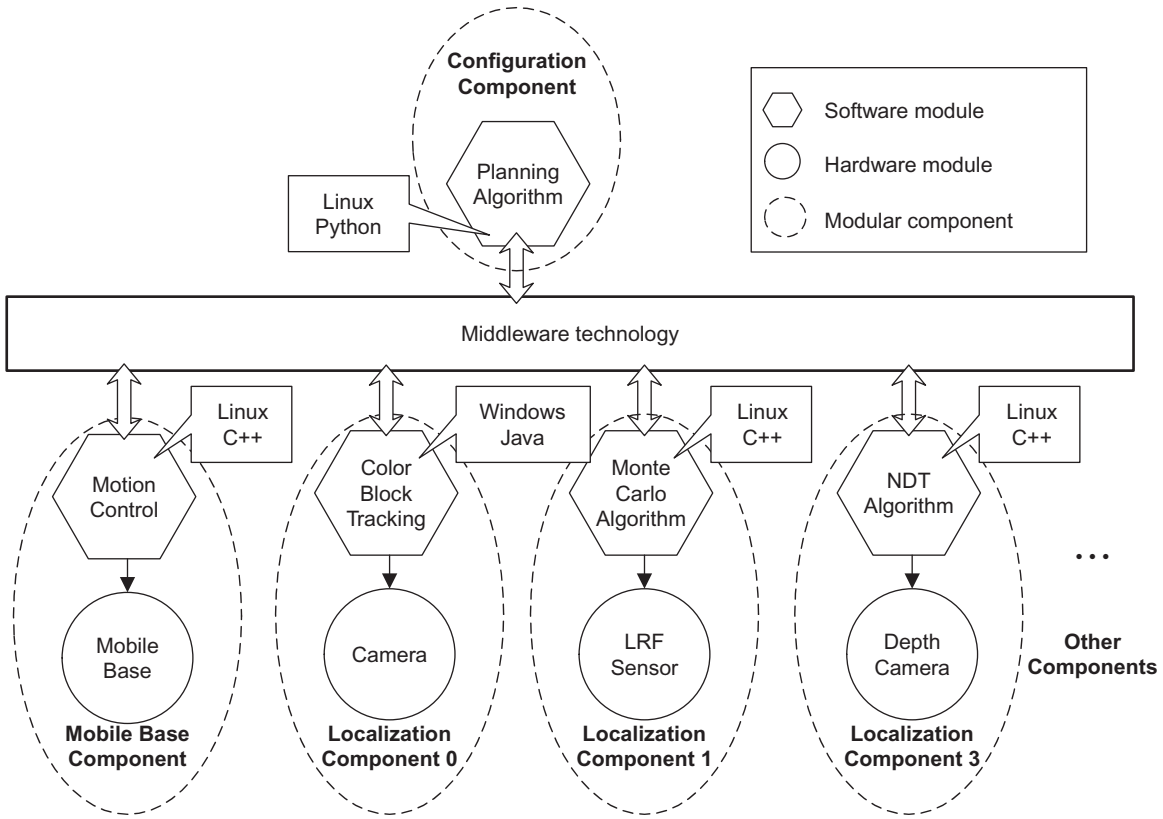


Figure 2 The hardware platforms in this paper

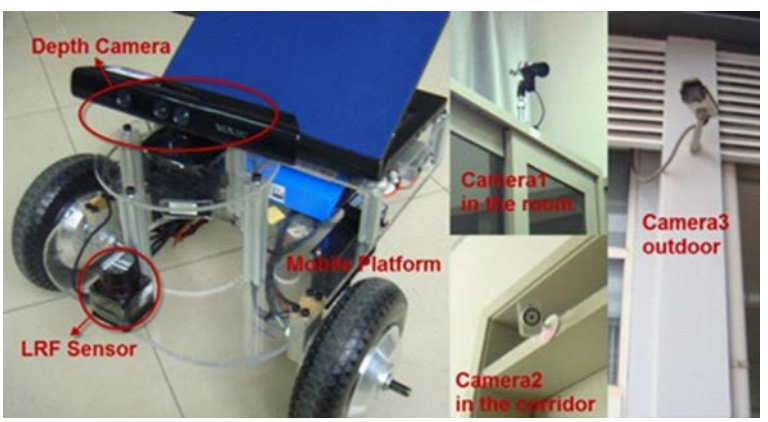
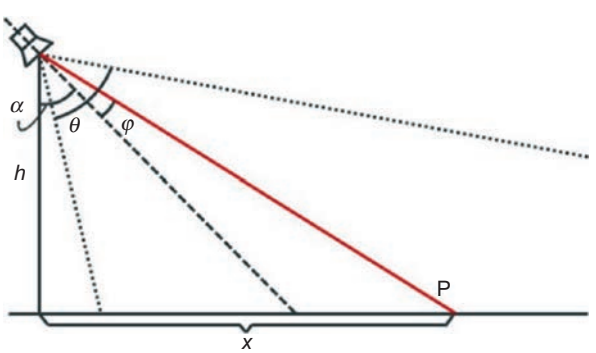


Figure 3 The environmental camera installation example



In practice, θ_0 is very small, and $\varphi + \alpha$ is smaller than $\pi/2$, therefore $\cot\theta_0 \gg \tan(\varphi + \alpha)$. It can be obtained:

$$D(\varphi) \approx \frac{\sec^2(\varphi + \alpha)}{\cot\theta_0} \cdot h \propto \sec^2(\varphi + \alpha) \tag{4}$$

If the width of the color mark is denoted by w, the pixel number of the color mark is given by:

$$n(\varphi) = \frac{w}{D} \tag{5}$$

Intuitively, the recognition accuracy is proportional to the color mark size in the image. It is defined as:

$$\Phi(\varphi) = \frac{n}{n_{\max}} \in [0,1], \text{ where } n_{\max} = n(\varphi_{\min}) = n\left(-\frac{\theta}{2}\right) \quad (6)$$

Supposing the recognition algorithm has an error of in $\pm k$ pixels, the actual error at point P is:

$$E(\varphi) = k \cdot D \quad (7)$$

The localization satisfaction S is inversely proportional to the localization error:

$$S(\varphi) = \frac{1}{E+1} \in [0,1] \quad (8)$$

The localization reliability R_c is the product of the recognition accuracy Φ and the satisfaction S:

$$\begin{aligned} R_c(\varphi) = \Phi \cdot S &= \frac{n}{n(-(\theta/2))(kD+1)} \\ &\approx \frac{w}{n(-(\theta/2))(kD^2+D)} \end{aligned} \quad (9)$$

From the actual parameters of the indoor camera, $\alpha = 40°$, $h = 2$ m, $N = 900$, $\theta = 58°$, the reliability $R_c(\varphi)$ is plotted in Figure 4.

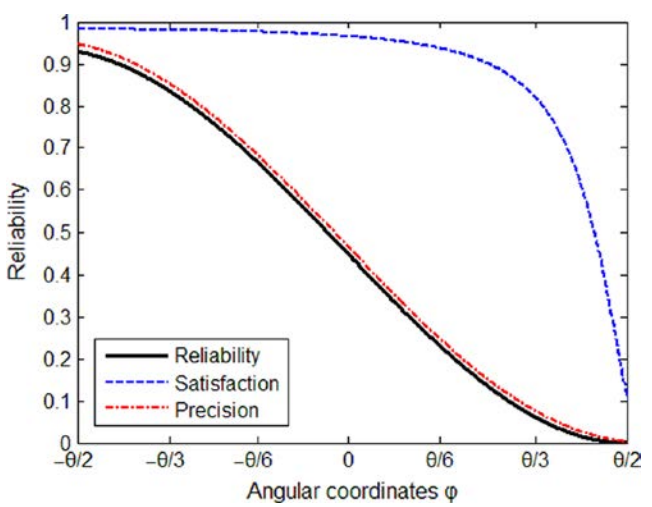
2.2 LRF sensor and odometer based approach

The LRF sensor is attached to the front of the mobile robot. It provides the 2D point cloud data of the environment. We employ the Monte Carlo localization algorithm, and implement it on the basis of the AMCL package (Gerkey, 2011; Thrun *et al.*, 2005) of robot operating system (ROS). This package performs the adaptive Monte Carlo localization approach for a robot moving with a 2D map. It runs at the frequency of 10 Hz, and has an average error of 15 cm in our environment. This algorithm has been very comprehensively studied and widely used, so in-depth analysis is omitted.

This approach takes in an offline built map and online laser scan data, and outputs pose estimates and the covariance matrix C, which is 6 by 6 in dimension:

$$C = (c_{ij})_{6\times 6} \quad i,j \in \{x, y, z, \alpha, \beta, \gamma\} \quad (10)$$

Figure 4 Reliability of environmental camera



This covariance matrix C is consisted of the covariance between six parameters, these are translation measured along x-, y-, z-axis and the rotation also measured about the XYZ-axis. Since our robot only moves on a 2D horizontal plane and there is a compass accounting for the rotation about z-axis, the localization error is defined as:

$$E = (c_{xx} + c_{yy})^{1/2} \quad (11)$$

where c_{xx} and c_{yy} are obtained from covariance matrix, and denote the square of average positioning error along X- and Y-axis, respectively.

In a manner similar to equation (8), the localization reliability for the LRF sensor is defined as:

$$R_l = \frac{1}{E+1} \in [0,1] \quad (12)$$

2.3 Depth camera and odometer based approach

The depth camera used in this paper is the Kinect sensor from Microsoft®. This device combines the information from a depth sensor and a standard RGB camera. In this study, only the point cloud generated by the depth sensor is used. The data is matched to an existing map using 3D normal-distributions transform (NDT) algorithm (Magnusson, 2009). This algorithm first applies the normal distribution model to represent the point cloud in a compact way, and then applies standard numerical optimization methods to maximize a likelihood function, so as to minimize the distance between corresponding points.

The purpose of localization is to obtain the position and orientation of the fresh point cloud with respect to the map point cloud. In the 3D space, a transformation matrix can be encoded to represent the 3D pose of a point cloud:

$$T = \begin{bmatrix} \cos\alpha\cos\beta & -\sin\alpha\cos\beta & \sin\beta & t_x \\ \sin\alpha\cos\gamma+\cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\gamma+\sin\alpha\sin\beta\sin\gamma & -\cos\beta\sin\gamma & t_y \\ \sin\alpha\cos\gamma-\cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\gamma-\sin\alpha\sin\beta\sin\gamma & \cos\beta\cos\gamma & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

Matrix T includes six independent variables: α, β, γ, t_x, t_y, t_z. Wherein α, β, γ represents the rotation around XYZ-axis, t_x, t_y, t_z represented the translation along the XYZ-axis.

Assuming that the existing map is represented as a point cloud $PCO = \{pco_1, pco_2, \ldots, pco_n\}$, while the fresh point cloud to be localized is represented as $PC = \{pc_1, pc_2, \ldots, pc_m\}$. A spatial transformation function Trs(PC, T) can moves a point cloud PC in space by the transformation matrix T, and the transformed point cloud is expressed as $PCT = \{pct_1, pct_2, \ldots, pct_m\}$, then:

$$PCT = \mathrm{Trs}(PC, T) = T \times PC \quad (14)$$

We treat the point cloud of the map PCO as a probability distribution D, assuming its probability distribution function (PDF) is Fd and its distribution parameters are contained in the matrix T. PCT would be a sample from PCO containing m sample values. The likelihood function $L(PC, T)$ is constructed to represent the probability of this sample:

$$\begin{aligned} \mathrm{L}(PC, T) &= \mathrm{Fd}(pct_1, pct_2, \ldots, pct_m | T) \\ &= \mathrm{P}(pct_1, pct_2, \ldots, pct_m) = \sum(pc_i, T) \end{aligned} \quad (15)$$

Since *PC* is known and constant, the only variable in likelihood function *L* would be matrix *T*. The maximum value of likelihood function L denotes that *PC* is most likely to be a sample from the map point cloud *PCO*, which also means that the localization error is minimal. To sum up, the best pose matrix *T* should be the one that maximizes the likelihood function.

From the above we can learn that PDF is the key to building likelihood function. A PDF is most suitable when it can locally captures the structure of the surface points robustly. The PDF chosen for this paper is expressed as:

$$\mathrm{P}(x) = \frac{1}{(2\pi)^{3/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^{\mathrm{T}}\Sigma^{-1}(x-\mu)\right) \quad (16)$$

where the mean vector μ and covariance matrix Σ are the characteristic parameters of the point cloud. The mean vector indicates the position where the point cloud is located, while the covariance matrix reflects overall distribution trend of point cloud. We employ the Newton optimization algorithm to search for the parameter T that optimizes $L(T)$. The odometer information was also recorded to initialize the estimate for the optimization algorithm.

The robot location can be easily deduced from the resulting translation matrix *T*. Furthermore, a localization score is obtained by summing the probabilities of normal distributions of all points *PC* with parameter *T*:

$$score = \sum_{i=0}^{m} \exp\left(-\frac{(pct_i - \mu)^{\mathrm{T}}\Sigma^{-1}(pct_i - \mu)}{2}\right) \quad (17)$$

We conclude the correspondence between the *score* and localization error *E* through a large number of experiments, and establish a lookup table.

Same as above, the reliability for the depth camera is represented as:

$$R_d = \frac{1}{E+1} \in [0,1] \quad (18)$$

Since the global map is built offline, the probability distribution *D* of the point cloud *PCO* can be calculated in advance. A lookup table is built to accelerate the online localization process, so that finding the corresponding normal distribution involves a simple lookup in the grid of the NDT. Furthermore, the point cloud to be located is rarefied to improve the efficiency. The point number is reduced to 5,000, and the localization rate is increased to 30 fps.

3. Modularization based on RTC

As mentioned above, the localization components are highly heterogeneous with respect to platform such as operating system, programming language and communication media. Thus, the middleware is employed to generalize the components into uniform standardized abstraction, which enables the dynamic communication and cooperation between any two of the modules. Figure 5 explains how middleware transparentizes the hardware and software platform, and offers standardized access to the robotic devices. Furthermore, it simplifies the developing process in that everyone can focus on developing his own module as long as compliance with the same port configuration. Consequently, the system is easy to modify and expand due to its highly modular characteristics.

We implement our system based on RT middleware owing to its two functionalities:

1 The data ports and service ports for data exchange and service invoking.
2 Component execution context for lifecycle management.

The ports are categorized as data ports and service ports. The data port is responsible for the continuous exchange of data. It performs in the data-flow style. Each component can have any number of data InPorts and OutPorts. A data OutPort sends the data to a corresponding InPort which receives the data. The service port provides the command based communication. It is executed in the client-server style. The component with a service provider, offering a set of services, listens for requests upon those services. The component with a service consumer, desiring that a service be performed, sends a request to the service provider via a connector. The service consumer and provider are called as interface. The port does not provide any functionality for data or command communication. Communication between components is actually performed by service interfaces. A port can associate functionally related service interfaces of any number and any direction. This means that one service port can provide multi-functionalities with bidirectional communication. Figure 6(a) shows the simplified UML model of the RT-component.

Figure 7 shows the seven components in our system. The five localization components are encapsulated into a unified form. They have one data OutPort for sending the robot location (x_t, y_t, θ_t), which denotes the coordinates and orientation in the world coordinates. Additionally they have one service port with two "provided interfaces", for the component's current state and the localization reliability, respectively. The mobile robot component is designed with one data InPort, which receives the location information, and one service interface, providing the component's state. The localization modules share unified port properties, as a result, the mobile platform module has the capacity to receive the data from any of the localization module using the same piece of code. The configuration module has two interfaces which are attached to a service port. It reads the localization components' reliability through the service port in real-time, chooses the optimal one, and connects the localization component to the robot component according to the situation.

As mentioned above, every component has its lifecycle, and its current state. Figure 6(b) briefly explains the execution states and their transitions. When a component is started, it is in the ready state, in which it has completed the initialization process. And then the component can be activated to execute the core algorithm repeatedly. It remains in the active state until it is deactivated or an error occurs. When an error occurs in the core algorithm, a reset can be performed and the component will shift back to the ready state (Table I).

4. Automatic switching algorithm

The configuration component acts like a supervisor in our system. It mainly performs two tasks:

1 Real-time monitoring the components' health status. Once the execution error occurs, the current configuration will be changed. At the same time, the problem component will be restarted.
2 Improving the reliability of the robot localization. Choosing the most reliable source for the mobile robot localization data.

Figure 5 Conceptual model of middleware

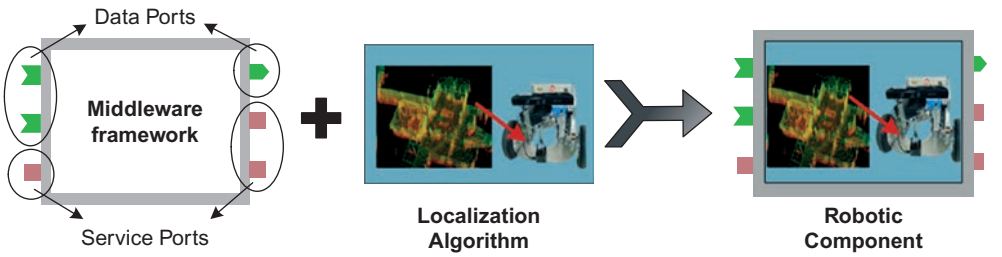


Figure 6 (a) Simplified UML component model and (b) component state of its lifecycle

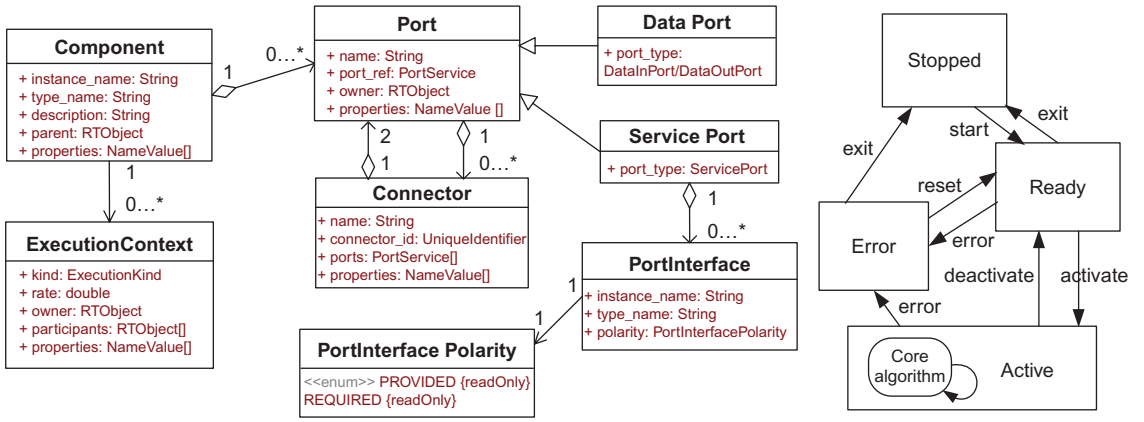


Figure 7 Components developed under middleware framework for this study

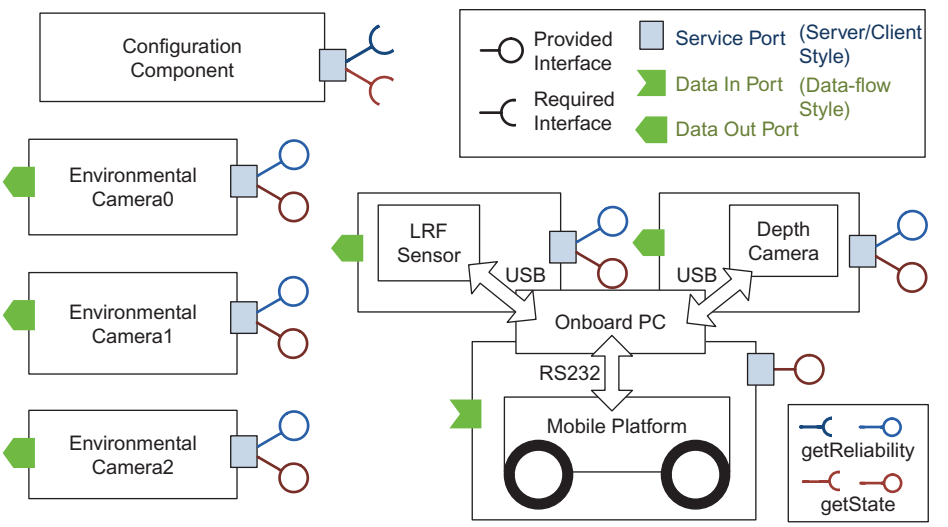


Table I Port definition and description of the components

Component	Port type	Interface	Data type	Description
Localization (all of the three kinds of components)	DataOutPort	LocOut	RoboLoc	Send component location
	ServicePort	getState	RoboState	Send component state
		getReliability	Float	Send localization reliability
Mobile platform	DataInPort	LocIn	RoboLoc	Receive component location
	ServicePort	getState	RoboState	Send component state
Configuration	ServicePort	getState	RoboState	Receive component state
		getReliability	Float	Get localization reliability

For the purpose of switching the localization component, it is convenient and intuitive to apply the greedy algorithm, that is, always choose the component with the highest reliability. This straightforward method has its drawback that if the reliabilities of two components are close to each other during a certain executing process, the configuration will change frequently between these two components. Therefore, we introduce a changing punishment P to the original greedy algorithm. Suppose the reliability of the current localization component is R_c, another one's is R_k. The component will be switched only if $R_k - P > R_c$. The configuration pseudo-code is presented below:

```
Switching Algorithm
while execution not terminated
    for each localization component C_i
      S_i ← getStatus(C_i)
      R_i ← getReliability(C_i)
      if S_i = = ERROR, then R_i ← 0 endif
    endfor
    R_max ← Max_{i=1,2,...,n} R_i
    if R_max − P > R_c, then
      Disconnect(C_c)
      Connect(C_max)
    Endif
End
```

where, $R \in (0,1)$ is the reliability obtained by the approach described in Section 2. The selection policy of P is as follow:

- For the fluency of the localization process, P is chosen to be as large as possible to avoid the oscillation between localization methods.
- For the localization accuracy, P is chosen to be as small as possible to avoid robot lost situation.

We implemented an experiment in which we drove the robot for 30 minutes and collected the data from all the components. Different punishment values were applied to the data. The switching times and the average localization error are calculated and illustrated in Figure 8. Based on the above two principles, we suggest to use $P \in [0.1, 0.17]$.

5. Experiments and results

The robot was set in a large and complicated environment, consisting of an indoor room, a corridor, a big hall and an outdoor square (Figure 10(a)). In the experiment, the scenes were deliberately chosen such that no single localization method was adequate. Further, some interference was also added such as changing some objects' position after building the map and having people moving around the robot, etc. The experiments proved no individual method had the capacity to successfully complete the entire task by itself.

As discussed above, each module is developed into RTC module. The three environmental cameras are implemented in Java on Windows. The three components on the mobile robot are developed in C++ on Linux. And the planning component is written in Python on Linux. The heterogeneous components can be connected seamlessly thanks to the middleware technology that enables the communication and collaboration in real-time. The planner performs a supervisory role in the manner we talked in Section 4. It first reads the reliability and state of each localization component from the service port, and then connects the data port of the mobile platform component to a favorable localization component. The system diagram is detailed in Figure 9. The arrowed lines starting from service provider to service consumer, transfer the states and reliabilities to the planner. The dash lines connecting the data InPort and OutPort, transfer the localization data. Upon analysis it becomes obvious that one can add any number of extra localization components to the system without any undue concern for maintaining and modifying the existing components.

We established both 2D and 3D maps offline (Figure 10(l)), and well installed the environmental cameras. The system developed in this paper focuses on the robustness in robot localization. In order to test the robustness, we have made a large number of long time experiments. The robot was made to walk along different routes by the remote control with the speed set to 0.5 m/s. As Figure 10 shows, it moved in a variety of scenes and switched the localization methods according to the situation. After 30 experiments each lasting about 30 minutes, it was found that only one lost situation occurred. For comparison, we also did a number of short-term experiments using a single localization method. The resulting statistic data is shown in Table II.

The table indicates that the LRF sensor and odometer based approach, as well as the depth camera and odometer based one, do not have good performance in the large-scale complicated environment. Due to the effective range of the LRF sensor being limited to 5 m, it always fails in the corridor.

Figure 8 Switch number and localization error with different changing punishment P

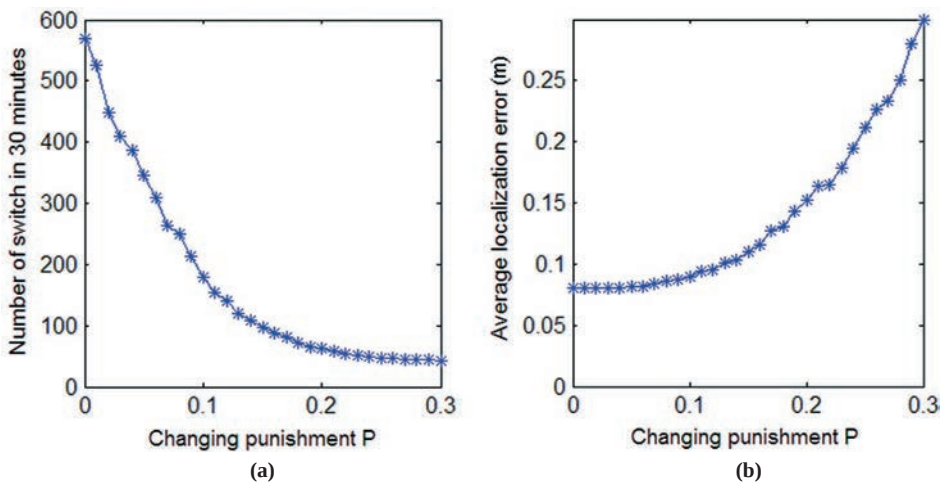


Figure 9 The system diagram of the experiment

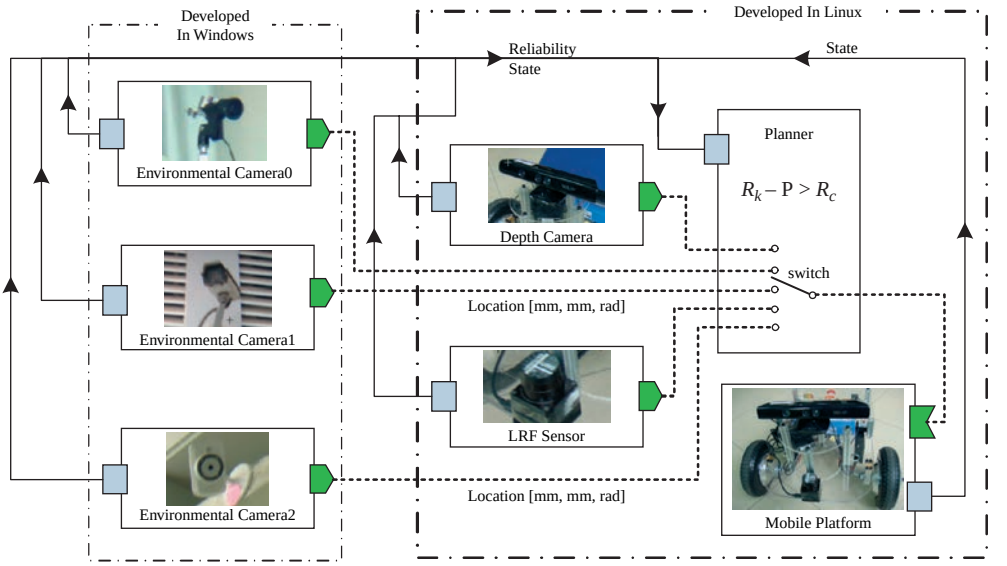
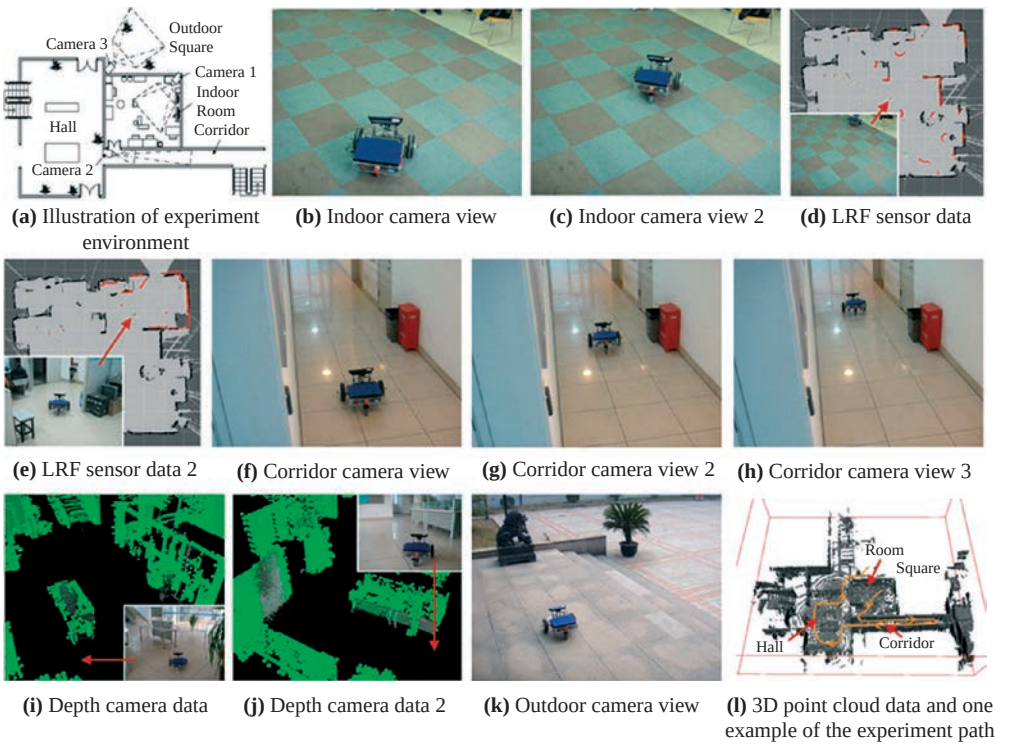


Figure 10 Experiment scenes and results



(a) Illustration of experiment environment
(b) Indoor camera view
(c) Indoor camera view 2
(d) LRF sensor data
(e) LRF sensor data 2
(f) Corridor camera view
(g) Corridor camera view 2
(h) Corridor camera view 3
(i) Depth camera data
(j) Depth camera data 2
(k) Outdoor camera view
(l) 3D point cloud data and one example of the experiment path

Table II The statistic results and comparison of the localization methods

	LRF sensor	Depth camera	RTC based approach
Total time (min)	60	60	900
Walking distance (m)	1,307	1,348	19,650
Number of lost situation	19	15	1
Average switch delay (s)	–	–	0.135
Average accuracy (m)	0.145	0.227	0.089
Average reliability	0.875	0.812	0.913

Depth camera's performance is also poor in the corridor. Its precision is vulnerable to the interference of close obstacles because of its narrow visual angle. It is clear that the collaboration of multi-robotic components significantly improves the robustness and reliability. In our long-term experiments up to 900 minutes, only one lost situation occurred, and this happened after prolonged human interference. In addition, the overall accuracy was significantly improved on account of the integration of various components' strongpoints. Making them compensate the deficiency of each other proved to be a good solution.

We verified the influence of two kinds of time delay on localization. One which is called the switch delay, is brought by switching the data port and the other is the communication delay on account of the network transmission. During a switching process, it takes an average of 135 ms for the data port channel to switch from the previous localization module to a new one. The impact is subtle, for the reason that the port does not switch frequently (typically switching happens about three to four times per minute). On the other hand, the effect of communication delay on localization is also very small attributed to the limited data flow. According to our experiments, it takes less than 10 ms through internet transmission from environmental camera to mobile platform, and less than 1 ms from LRF and depth camera. The latter time is shorter because those components are implemented on the same computer. The velocity is 0.5 m/s, so the communication delay brings the localization error less than 0.5 cm, which may be considered negligible.

Figure 11 shows the reliability curves in the first 70 seconds of one of the experiments. The robot followed the route marked in Figure 10(l). The planning system changed the configuration for six times in the whole process. At the beginning, the environmental camera was employed. The result of LRF based approach is poor, because it had not converged. When the robot moved far from the camera, its reliability decreased. The first switch occurred to utilize the depth camera component. Subsequently, although the LRF component returned better results than depth camera did, the system did not change its configuration because the superiority did not overcome the changing punishment. The robot then moved to the corridor and changed to make use of another environmental camera until the third switch occurred upon reaching the end of the corridor in Figure 10(h). The robot returned from the corridor and entered the big hall while switching to the LRF sensor. At the fifth switch, some people passed the hall with the result that the robot was seriously affected and the localization component was switched to the depth camera. The last switch happened because the robot reached the door and was about to move to the out square, under this conditions the Kinent's performance fell as a result, the LRF sensor took over the localization task again.

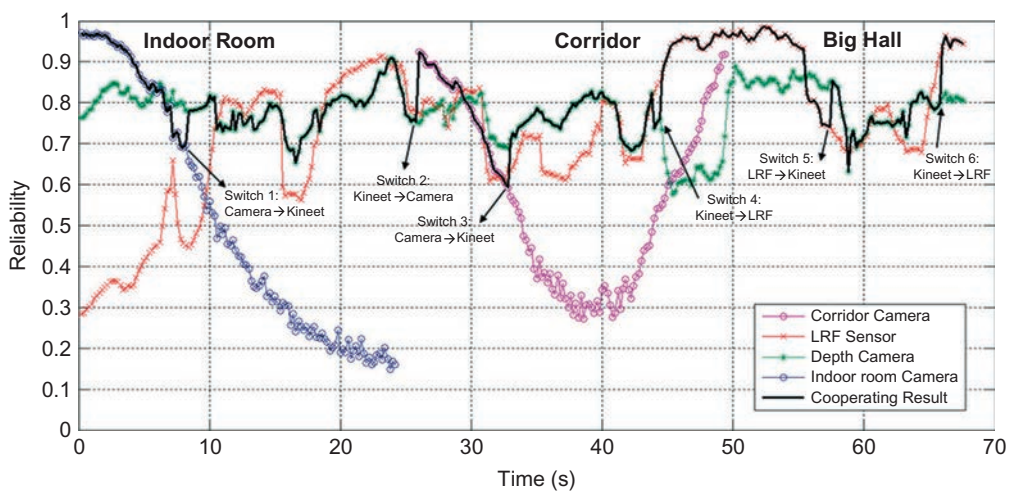
It is obvious that this approach has combined the advantages of all the components, and avoided their shortcomings. To maximize this effect, the off-board components are supposed to be set in other's weakness position. For example, the environmental camera should work at the area where the onboard methods often fail. And other localization techniques such as RFID (Cicirelli *et al.*, 2012) or landmark (Lee, 2009) based approaches can also join in at the appropriate location to make the system stronger.

6. Conclusions and future works

A novel localization method in complicated environment based on modular development, taking advantage of communication and cooperation of heterogeneous robotic components has been presented and successfully implemented. The middleware technology which is required to achieve the long-term coordination and easily expansion of these components has been discussed. Experiments based on seven components which are three environmental cameras, a LRF sensor, a depth camera, a mobile base and a planning module, achieving a robust localization system, were presented and applied successfully. One deficiency exists arising from the lack of a unified model of the reliability for all the localization components due to their high diversity. We give the examples on how to model the reliability on the basis of localization accuracy. It has been proved that our work achieves the desirable result of improving the robustness and accuracy of the localization system.

Currently further efforts are being made to implementing a data fusion approach that rather than only chooses one localization method each time, but combines them by minimizing the likelihood function of their joint distribution function. We plan to compare the fusion results with the results in this paper in the future. What is more, the methods and

Figure 11 The reliability of the localization components in one of the experiments



technologies presented in this paper are not limited to the robot localization problem, but aiming at more general heterogeneous robotic components and tasks. We have been working on a grasping system based on similar methodologies. And advanced planning algorithms are under investigation for the better automation and higher intelligence.

References

Ando, N., Suehiro, T. and Kotoku, T. (2008), "A software platform for component based RT-system development: openRTM-aist", *Simulation, Modeling, and Programming for Autonomous Robots*, pp. 87-98.

Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T. and Yoon, W.-K. (2005), "RT-middleware: distributed component middleware for RT (robot technology)", *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3933-3938.

Biswas, J. and Veloso, M. (2012), "Depth camera based indoor mobile robot localization and navigation", *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1697-1702.

Chen, X., Huang, Y., Liu, G. and Guo, J. (2008), "Localization algorithm of mobile robot based on single vision and laser radar", *7th World Congress on Intelligent Control and Automation*, pp. 7667-7671.

Chenavier, F. and Crowley, J.L. (1992), "Position estimation for a mobile robot using vision and odometry", *Proceedings of 1992 IEEE International Conference on Robotics and Automation*, pp. 2588-2593.

Cicirelli, G., Milella, A. and Di Paola, D. (2012), "RFID tag localization by using adaptive neuro-fuzzy inference for mobile robot applications", *Industrial Robot: An International Journal*, Vol. 39 No. 4, pp. 340-348.

Collett, T.H.J., MacDonald, B.A. and Gerkey, B.P. (2005), "Player 2.0: toward a practical robot programming framework", *Proceedings of the Australasian Conference on Robotics and Automation* (*ACRA 2005*).

Davison, A.J., Reid, I.D., Molton, N.D. and Stasse, O. (2007), "MonoSLAM: real-time single camera SLAM", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29 No. 6, pp. 1052-1067.

Delalloche, L., Pegard, C. and Marhic, B. (2001), "A dynamic omnidirectional localization system for mobile robot navigation", *International Journal of Robotics & Automation*, Vol. 16 No. 1, pp. 26-33.

Drocourt, C., Delahoche, L., Pegard, C. and Clerentin, A. (1999), "Mobile robot localization based on an omnidirectional stereoscopic vision perception system", *Proceedings of 1999 IEEE International Conference on Robotics and Automation*, pp. 1329-1334.

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D. and Burgard, W. (2012), "An evaluation of the RGB-D SLAM system", *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1691-1696.

Fox, D., Burgard, W. and Thrun, S. (1998), "Active Markov localization for mobile robots", *Robotics and Autonomous Systems*, Vol. 25 No. 3, pp. 195-207.

Fox, D., Thrun, S., Burgard, W. and Dellaert, F. (2001), "Particle filters for mobile robot localization", *Sequential Monte Carlo Methods in Practice*, pp. 401-428.

Gerkey, B.P. (2011), "AMCL", available at: www.ros.org/wiki/amcl

Huang, J., Supaongprapa, T., Terakura, I., Wang, F., Ohnishi, N. and Sugie, N. (1999), "A model-based sound localization system and its application to robot navigation", *Robotics and Autonomous Systems*, Vol. 27 No. 4, pp. 199-209.

Jensfelt, P. and Kristensen, S. (2001), "Active global localization for a mobile robot using multiple hypothesis tracking", *IEEE Transactions on Robotics and Automation*, Vol. 17 No. 5, pp. 748-760.

Lee, S. (2009), "Use of infrared light reflecting landmarks for localization", *Industrial Robot: An International Journal*, Vol. 36 No. 2, pp. 138-145.

Magnusson, M. (2009), "The three-dimensional normal-distributions transform", doctoral dissertation, Örebro University, Örebro.

Meng, G., Wanli, L. and Zhankui, W. (2011), "Robot navigation based on multi-sensor data fusion", *Second International Conference on Digital Manufacturing and Automation* (*ICDMA*), pp. 1063-1066.

Mohamed, N., Al-Jaroodi, J. and Jawhar, I. (2009), "A review of middleware for networked robots", *International Journal of Computer Science and Network Security*, Vol. 9 No. 5, pp. 139-148.

Murphy, A.L., Picco, G.P. and Roman, G.-C. (2006), "LIME: a coordination model and middleware supporting mobility of hosts and agents", *ACM Transactions on Software Engineering and Methodology* (*TOSEM*), Vol. 15 No. 3, pp. 279-328.

Rekleitis, I., Meger, D. and Dudek, G. (2006), "Simultaneous planning, localization, and mapping in a camera sensor network", *Robotics and Autonomous Systems*, Vol. 54 No. 11, pp. 921-932.

Skrzypczynski, P. (2012), "Laser scan matching for self-localization of a walking robot in man-made environments", *Industrial Robot: An International Journal*, Vol. 39 No. 3, pp. 242-250.

Teslić, L., Škrjanc, I. and Klančar, G. (2010), "Using a LRF sensor in the Kalman-filtering-based localization of a mobile robot", *ISA Transactions*, Vol. 49 No. 1, pp. 145-153.

Thrun, S., Burgard, W. and Fox, D. (2005), *Probabilistic Robotics*, MIT Press, Cambridge, MA.

Thrun, S., Fox, D. and Burgard, W. (2000), "Monte Carlo localization with mixture proposal distribution", *Proceedings of the National Conference on Artificial Intelligence*, pp. 859-865.

Thrun, S., Fox, D., Burgard, W. and Dellaert, F. (2001), "Robust Monte Carlo localization for mobile robots", *Artificial Intelligence*, Vol. 128 Nos 1/2, pp. 99-141.

Utz, H., Sablatnog, S., Enderle, S. and Kraetzschmar, G. (2002), "Miro-middleware for mobile robot applications", *IEEE Transactions on Robotics and Automation*, Vol. 18 No. 4, pp. 493-497.

Xiong, C., Han, D. and Xiong, Y. (2009), "An integrated localization system for robots in underground environments", *Industrial Robot: An International Journal*, Vol. 36 No. 3, pp. 221-229.

Corresponding author

Wenshan Wang can be contacted at: amigo@sjtu.edu.cn