The 6th Asian Control
July 18-21, 2006
Bali, Indonesia

Mobile Robot Control Architecture for Reflexive Avoidance of Moving Obstacles

Won-Jong Sohn*, Keum-Shik Hong*, Jae-Bok Song, and Kyung-Hyun Choi** [=]

* School of Mechanical Engineering, Pusan National University, Busan, Korea.
e-mail: {wjson, kshong}@pusan.ac.kr

** Department of Mechanical Engineering, Korea University, Seoul, Korea.
e-mail: jbsong@korea.ac.kr

[=] Department of Mechatronics Engineering, Cheju National University, Korea
e-mail: khchoi@cheju.ac.kr

Abstract

In this paper, obstacle avoidance in the reflexive layer of a hybrid deliberative/reflexive control architecture is discussed. The layer consists of resources, actions, an action coordinator and motion controller. Individual resources work independently. To improve the real-time characteristics, a set of simplified actions and an action coordinator are designed using functions provided in the RTAI (Real-Time Application Interface for Linux) environment. Also, an obstacle avoidance algorithm using the geometric relationship between the robot and obstacles is proposed. The Kalman filtering method is debated to get more exact impomation. The effectiveness and the real-time characteristic of the proposed reflexive mechanism are verified through experimental results using organized scenarios.

1 Introduction

The necessity for productivity improvement has brought the development of industrial robots, and the desire of human being for convenience has intrigued the appearance of human-friendly robots including a service robot for disabled people and a silver robot for elderly people. It can be said that the coexistence of human and robots in the same environment is not far away. As the human's expectation for robot grows, the hardware of robots becomes complicated and more diverse sensors are used. Most of all, the system integration of various software as well as hardware becomes more important. For this purpose, an architecture that allows an easy integration of diverse software is demanded. After selecting an operation system that can be appropriately applied depending on needs, a software framework should be designed according to its characteristics and required performance. Such a framework, which has defined standards, expansion and requirements, is a control architecture. As shown in the example below, it is difficult for existing control architectures to satisfy the current performance requirements, and therefore it is necessary to introduce a new concept control architecture to satisfy the various functions required.

In the early 1980s, the SPA structure of batch processing was the prevailing structure in control architecture research. The shortcomings of such a method lied in the fact that a great deal of knowledge about the working environment is necessary in advance to design a stable planning part and that it is difficult to cope effectively with uncertain and unpredictable actual environments.

The control architectures in the 1980s were characterized by layer control architectures that included Brooks' subsumption architecture[4]. The shortcomings of subsumption architecture lies in the fact that it failed to put forward mechanisms that can efficiently deal with a growing number of layers. Not only to complement the above shortcomings but also to meet the varied performance requirements of complex hardware, studies have been conducted from the early 1990s on layer control architectures, behavior-based control architectures, and control architectures combining competitive coordinators with cooperative coordinators[9,10,12]. This architecture is layered so that it may establish appropriate action plans by interpreting given tasks and systematically generate processes according to action plans, in addition to managing them in a systematic manner.

Representative hybrid control architectures include 4D/RCS[2] developed by NIST to apply to unmanned military vehicles and CLARAty[12] jointly developed by NASA, California Institute of Technology, and Carnegie Mellon University to apply to a Mars exploration robot. In Korea, KIST has developed the Tripodal Schematic Architecture(TSA, [8]) and applied it to a service robot and KAIST has developed a control architecture for a semi-autonomous mobile wheelchair[9]. The TSA consists of a layered functionality diagram that expresses the overall structure and information flows, a class diagram that conducts programming by function to realize components, and a configuration diagram that executes orders and correct errors using a petri-net.

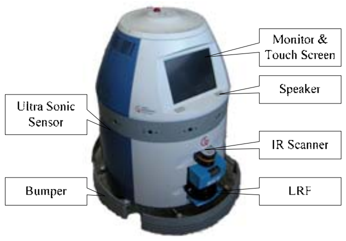


Figure 1: The used Mobile Robot (21st Century Frontier R&D Program)

But most of these architectures have the shortcoming that all situations that can occur when a robot moves should be predicted and constructed in a database in order for the robot to autonomously move under the control architecture. The examination of the control architectures of the reflexive layers in the various control architectures presented above shows that each factor was modeled with an object-oriented approach to show excellent recyclability and flexibility but the relations of data flows are not clear. It also shows that the architectures focused on the realization of algorithms using various sensor systems and thus they fail to realize the characteristics of a reflexive layer, in addition to having insufficient reference to real-time realization.

Accordingly, this study selects and designs structural elements of a reflexive layer in a control architecture that integrates and controls a robot and explores ways to guarantee real-time realization. In addition, this study proposes a mechanism of real-time traveling control architecture to secure the movement and control stability of an autonomous mobile robot as a base for the autonomous functions of a mobile robot. As the core of a traveling control architecture, this study also designs actions for each function to which various algorithms will be applied for autonomous traveling.

2 System Realization

2.1 Hardware Architecture

An autonomous robot is a mobile robot that is capable of self-judgment and independent navigation in an unknown environment. Figure 1 shows the mobile robot used in this study, which was developed by the 21st Century Frontier R&D Program, KIST, Korea. The robot consists of 12 V two batteries, two driving wheels (left and right), and two auxiliary wheels (front and rear).

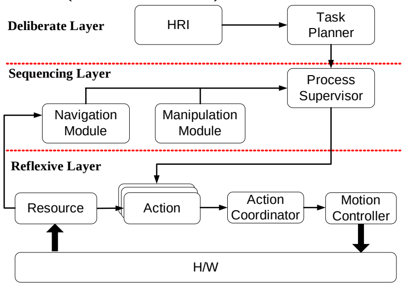


Figure 2: The adopted three-layer control architecture.

2.2 Software Structure

It is important to improve the mechanical architecture of a robot to improve its safety and add various functions, but it is also necessary to construct various types of software factors by adding real-time control systems and design and realize real-time software architectures that can efficiently distribute system resources by precisely defined data flows between components. Accordingly, real-time tasks like motor control, sensing and obstacle avoidance are required for autonomous system traveling. Tasks have different cycles, priorities, performance time and calculation amounts and require real-time execution in relation to sensor and motor driving. Therefore, it is indispensable to construct a software architecture that can efficiently manage and control them on a real-time basis. In this respect, this study uses RedHat Linux 9.0 patched with PTAI 3.1 to guarantee real-time execution.

3 Control Architectures of Reflexive Layer

Control architectures can be divided into deliberate control and reflexive control depending on whether or not the changes of the surrounding environment are considered in determining the robot behaviors. Figure 2 shows a control architecture adopted in this study. The control architecture has a 3-layer architecture which is widely studied by researchers in the robot field. Each of the layers has the following roles.

The Deliberative layer plays a role in changing human orders into mechanical language orders so that they can be carried out by the robot. It also takes the role of planning and dividing the tasks of the robot into the traveling part and the operational part. The Sequencing layer processes the information acquired from sensors and generates consecutive information to perform the work plans generated in the deliberative layer. The Navigation-Module of the sequencing layer uses delicate algorithms that take a lot of time and generates high-dimensional information from sensors including self position recognition, environmental map creation, movement course generation and stopover point calculation which are delivered to the RTAS of the reflexive layer. Lastly, the reflexive layer is the layer on which this study put its focus. The layer carries out orders from the upper classes. To guarantee the safety of humans and robots, the layer also responds to obstacles in real time in an optional environment and repeatedly performs simple computations periodically or non-periodically to control the movements of a robot. The reflexive layer consists of the Resource, Action, Action-Coordinator, Motion-Controller, and RTAS. It also uses a number of Actions and Action-Coordinators to generate natural movements. Now, let's take a look at each of the components.

3.1 Resource

The Resource represents a component that stores the data acquired from the sensors. Each sensor has a separate

Resource component and the stored information is provided to various Actions and superior components, if necessary.

3.2 Actions

The Action receives several pieces of information from various Resources and generates translation velocity and rotative angular velocity to move the robot. The Action is a component that realizes specific functions which the robot can combine and use in their autonomous traveling. Using the functions provided from the RTAI, it is realized as an RT-thread. The composition of the Action should be simplified to a maximum extent to minimize the time required for calculations. The actions of the reflexive layer are composed of the Goto that generates the robot's movement orders according to the orders from superior layers, the Avoid to evade obstacles, the Move needed to control velocities, and the EmergencyStop needed to stop in emergency.

3.2.1 Goto

The Goto represents an action to move the robot to the destination by receiving the points of stopover of the robot route from the Process-Supervisor. The points of stopover of the robot route from the current position to the target point are calculated in the Localizer and the Path-Planner of the Navigation-Module in the sequencing layer or they are calculated with the use of the information from the Encoder. The execution cycle of the Goto (20 ms) is faster than the update cycle for the current position of the robot in the Localizer, and it has excellent real-time hard characteristics. Therefore, it should use the traveling information from the Encoder until it receives the precise robot position information from the Localizer. Upon calculating the current position, the movement velocity and angular velocity are calculated and delivered to the Action-Coordinator in a consecutive manner. The results of [1] were adopted as the algorithm for the generation of the velocity and angular velocity and the robot's translational velocity (v) and rotative velocity (ω) are as follows.

$$v = k_3 \cos\rho\,,\ k_3 > 0\,, \tag{1}$$

$$\omega = k_4\rho + k_3\frac{\cos\rho\sin\rho}{\rho}(\rho + k_2\theta_d)\ ,k_2,\ k_3,\ k_4 > 0\,. \tag{2}$$

To apply this algorithm, it is necessary to generate the orders of the robot's translational and rotative velocities by setting the point of stopover two to three sections prior to an optional time t as the target point. This is because the actual robot should pass through several points of stopover to reach a final destination and, in this algorithm, the velocity becomes zero at all points of stopover.

3.2.2 Avoid

The Avoid of the reflexive layer refers to instinctive avoidance, and it is a behavior that can properly cope with a given situation. The algorithm of the Avoid presented in this study can generate the translational and rotative velocities for the robot's avoidance action in response to the distance, velocity and direction of the obstacle measured from the sensors. A variety of sensors can be used for the recognition of obstacles but only LRF was used in the current stage. Detailed algorithms will be explained in Sections 4 and 5.

3.2.3 Move

In general, a robot is controlled so that it may move from point to point, but the robot's position control method becomes ambiguous when performing tasks like object tracking or wall surface following. The Move has been introduced to solve such a problem.

3.2.4 EmergencyStop

The EmergencyStop means stopping a robot using a hardware motor brake in the Motion-Controller. The EmergencyStop becomes activated when the robot's bumper collides with an obstacle or when an obstacle comes within the emergency stop distance while the robot is in the process of avoiding an obstacle.

3.3 Action-Coordinator

By fusing the output of the Action in an effective manner, the Action-Coordinator delivers the generated movement orders to the Motion-Controller to generate a variety of soft movements. With the Action-Coordinator, a robot moves at an optional point of time with an Action or a combination of several Actions. In this study, the algorithm part was designed with a function so that it could be changed easily. As the current stage is a stage for performance evaluation, the algorithm was designed to change the Avoid when an obstacle comes within 80 cm, rather than adding complicated algorithms. Figure 3 shows a flow diagram of the Action-Coordinator which was designed to present an architecture that can satisfy the design requirements. The Action-Coordinator calculates the rotative velocities of each wheel with the final information to drive the robot.

3.4 Motion-Controller

The Motion-Controller is a component that actually delivers the control signals to the motion board. The Motion-Controller receives the velocity information of both wheels through the designated protocol from the Action-Coordinator, converts it into the type wanted by the actuator, and delivers it to the two wheels trough the real-time driver suitable for the interface used.

3.5 Real-Time Action Supervisor (RTAS)

The RTAS is a component of the reflexive layer and manages the Resource and behaviors. It offers a variety of hardware communication methods like robot controllers and sensor Resources that are lower components. As a part connecting the user space to the kernel space, the RTAS also designates the action mode of the Action to the Action-Coordinator and offers position and velocity information for the robot movement.

Before making an action to avoid an obstacle, a robot needs to clearly identify the dynamic status of the obstacle.

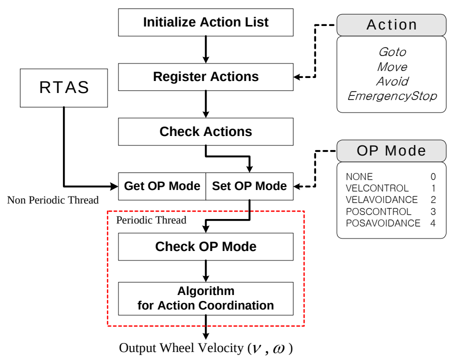


Figure 3: Data flow of the Action_Coordinator.

4 Obstacle Detection

This is because an obstacle that stands still does not become an object of consideration because it is recognized as an environment, but a moving obstacle can be avoided precisely only when the velocity and moving direction are predicted precisely. Accordingly, in this section, while guaranteeing real-time performance, the information on the surrounding environment will be acquired using the LRF among the various sensors mounted on the robot and the methods that can expel the obstacle within the information will be dealt with.

4.1 Obstacle Identification

The data acquired from the LRF can be expresses as:

$$LRF_k = \left\{ P_{k,i} = \begin{pmatrix} \alpha_{k,i} \\ d_{k,i} \end{pmatrix}, \quad i \in [0,360], \quad \Delta\alpha = 0.5^\circ \right\}$$

Of the 361 data obtained, objects existing within a certain range in the robot progressing direction will be detected. Here, k represents the k-th discrete time, α_i the i-th angle, and d_i the distance to the object obtained from that angle.

4.1.1 Segmentation

To recognize an obstacle, it is necessary to extract the obstacle information from the LRF data. The segmentation process begins with calculating the distance between the two points acquired consecutively. As shown in Figure 4, if the distance between the two succeeding points is the same as or smaller than the variable critical value (C0+C1), they will be classified as the same piece, and if the distance is larger than the critical value, they will be classified as different pieces.

4.1.2 Circularization with filtered three points

Of the data on an obstacle that are acquired by segmentation, the point nearest to the robot and the two end points are used to find the center of a circle supposing that the obstacle is circular. Circulation is a process necessary to consider the relativity of movements because the surface of the obstacle nearest to the robot at the k-th point of time is not the nearest part at the k+1th point of time.

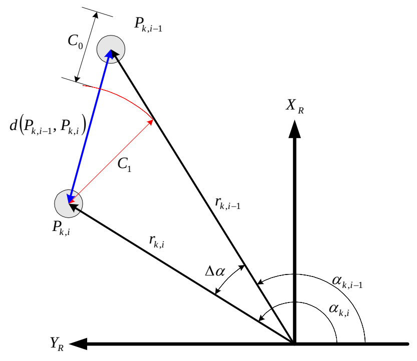


Figure 4: Segmentation

For circulation, when the robot exists at an optional point of time k , $P_{k,ini}$, $P_{k,near}$, and $P_{k,end}$ can be converted into the following rectangular coordinates.

$$P_{k,ini} = (x_{k,ini},\ y_{k,ini}), \quad where \quad x_{k,ini} = r_{k,ini} \times \cos\alpha_{k,ini}, \quad y_{k,ini} = r_{k,ini} \times \sin\alpha_{k,ini}, \tag{3}$$

$$P_{k,near} = (x_{k,near},\ y_{k,near}), \quad where \quad x_{k,near} = r_{k,near} \times \cos\alpha_{k,near}, \quad y_{k,near} = r_{k,near} \times \sin\alpha_{k,near}, \tag{4}$$

$$P_{k,end} = (x_{k,end},\ y_{k,end}), \quad where \quad x_{k,end} = r_{k,end} \times \cos\alpha_{k,end}, \quad y_{k,end} = r_{k,end} \times \sin\alpha_{k,end} \tag{5}$$

y_a and y_b, the two straight lines passing through the three points, can be obtained in the following way. m_a and m_b represent the slopes of each of the lines (Figure 5).

$$y_a = m_a\ (x - x_{k,ini}) + y_{k,ini}, \quad y_b = m_b\ (x - x_{k,near}) + y_{k,neari}, \tag{6}$$

$$m_a = \frac{y_{k,near} - y_{k,ini}}{x_{k,near} - x_{k,ini}}, \quad m_b = \frac{y_{k,end} - y_{k,near}}{x_{k,end} - x_{k,near}}. \tag{7}$$

Straight line y_a' which vertically bisects y_a and straight line y_b' which vertically bisects y_b can be expressed as follows.

$$\begin{aligned} y_a' &= -\frac{1}{m_a}\left(x - \frac{x_{k,ini} + x_{k,near}}{2}\right) + \frac{y_{k,ini} + y_{k,near}}{2}, \\ y_b' &= -\frac{1}{m_b}\left(x - \frac{x_{k,near} + x_{k,end}}{2}\right) + \frac{y_{k,near} + y_{k,end}}{2}. \end{aligned} \tag{8}$$

The position of the middle point of the obstacle exists at the point of intersection of the two straight lines and therefore the coordinates of axis x can be obtained as follows by calculating $y_a' = y_b'$.

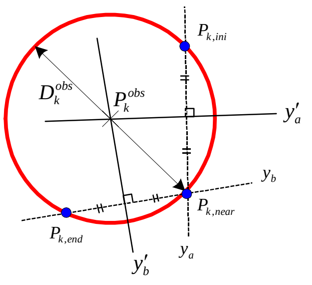


Figure 5: Circularization with filtered three points.

$$x_k^{obs} = \frac{m_a m_b \left(y_{k,ini} - y_{k,end}\right) + m_b \left(x_{k,ini} + x_{k,near}\right) - m_a \left(x_{k,near} + x_{k,end}\right)}{2\left(m_b - m_a\right)}. \quad (9)$$

The value of axis y can be obtained from $y'_a |_{x_k^{obs}} = y_k^{obs}$ (or $y'_b |_{x_k^{obs}} = y_k^{obs}$). When P_k^{obs} (the position of the calculated obstacle) was expressed as $\left\{x_k^{obs}, y_k^{obs}, r_k^{obs}\right\}$, the diameter of the circularized obstacle and the angle to the center of the robot can be expressed as follows:

$$D_k^{obs} = 2 \times \sqrt{\left(x_{k,near} - x_k^{obs}\right)^2 + \left(y_{k,near} - y_k^{obs}\right)^2}, \quad (10)$$

$$\theta_k^{obs} = \tan^{-1}\left(\frac{y_{k,near} - y_k^{obs}}{x_{k,near} - x_k^{obs}}\right). \quad (11)$$

4.2 Estimation of Position and Velocity of a Moving Object

The exact position of an obstacle can be estimated by integrating various noise information generated in the sensing process and the information centered on the obstacle obtained in the previous section. This is particularly an indispensable process in the case of a sensor with a lower degree of reliability. In the case of a moving obstacle, it is the information that can help predict the future route. When applying the Kalman Filter, ρ and θ are estimated respectively for a low-performance processor and to shorten the time in executing the program[15]. Figure 6 shows the concept of the Kalman Filter and (12) indicates the basic numerical expression of the Kalman Filter.

$$\begin{aligned} &\hat{x}_k(-) = \Phi_{k-1}\hat{x}_{k-1}(+), \\ &p_k(-) = \Phi_{k-1} p_{k-1}(+) \Phi_{k-1}^T + Q_{k-1}, \\ &\hat{x}_k(+) = \hat{x}_k(-) + K_k\left[y_k - H_k \hat{x}_k(-)\right], \quad (12) \\ &p_k(+) = \left(I - K_k H_k\right) p_k(-), \\ &K_k = p_k(-) H_k^T \left[H_k p_k(-) H_k^T + R_k\right]^{-1}. \end{aligned}$$

5 Collision Avoidance Algorithm

In evaluating the safety and traveling abilities of autonomous mobile technologies of an autonomous mobile robot, obstacle avoidance is one of the most basic and important technologies. Of the various sensor-based traveling technologies, this study deals with reflexive control.

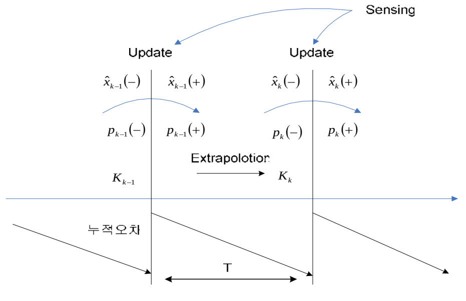


Figure 6: Concept of Kalman Filter algorithm.

The core of reflexive control is to avoid an obstacle in real time. To this end, simple algorithms, rather than complex ones, are more efficient for faster computations. In this section, the point nearest to the robot is located using the obstacle's coordinate, radius and origin obtained in the previous section. In addition, this section explains the process in which velocities and angular velocities are created though use of the algorithm that makes use of the geometric relations between the robot and the nearest point.

5.1 Conversion of the obstacle's direction and distance measured from the center of the LRF sensor to the center of the robot

The distance to the obstacle nearest to the robot can be obtained using the radius and distance between the center of the robot and the center of the circle obtained through circularization. But the relative coordinates of the obstacle obtained in this way are the data measured not from O_R, the center of the robot, but from the position of the LRF. As the robot is driven around O_R located on the same line as the driving wheel, it is necessary to move the relative coordinates of the obstacle to the center of the robot. As Figure 7 shows, using the information obtained through segmentation and circularization, the angle (α_i) and the distance (d_i) to the obstacle nearest to the center of the robot can be calculated. As in Figure 7(a, b), α' can be obtained in the following way by considering the case of $\alpha_i < \pi/2$ and the case of $\alpha_i > \pi/2$.

$$\alpha' = \begin{cases} \frac{\pi}{2} + \alpha_i, & 0 < \alpha_i \le \frac{\pi}{2} \\ 0 \quad , & \alpha_i = \frac{\pi}{2} \\ \frac{3\pi}{2} - \alpha_i, & \frac{\pi}{2} < \alpha_i < \pi, \end{cases} \quad (13)$$

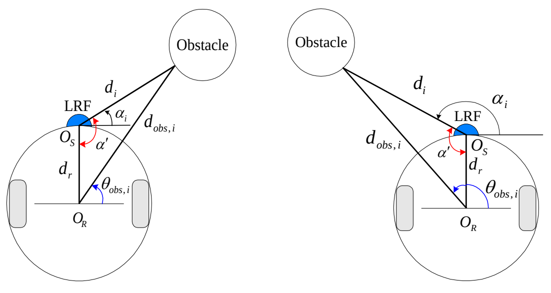
$$d_{obs,i} = \begin{cases} d_i + d_r & , \quad \alpha_i = \frac{\pi}{2} \\ \sqrt{d_i^{\,2} + d_r^{\,2} - 2\cos\alpha' d_i d_r} & , \quad otherwise, \end{cases} \quad (14)$$

$$\theta_{obs,i} = \begin{cases} \alpha_i & , \quad \alpha_i = \frac{\pi}{2} \\ \cos^{-1}\left(\frac{d_i^{\,2} - d_{obs,i}^{\,2} - d_r^{\,2}}{2 d_{obs,i} d_r}\right), & otherwise. \end{cases} \quad (15)$$

The distance ($d_{obs,i}$) between the center of the robot (O_R) and the obstacle can be obtained as in (14) by applying trigonometric function cosine law number 2. The angle between the center of the robot and the obstacle ($\theta_{obs,i}$) can be obtained by applying to (15) the distance to the obstacle obtained in (14), the distance between the center of the robot (O_R) and the center of the sensor (O_S), and the distance between the center of the sensor and the obstacle.

5.2 Autonomous mobile robot determines the moving direction

Figure 8 indicates a brief diagram of the obstacle's position information (angle and distance) obtained and the angle is measured counterclockwise. When the distance between the



(a) $\alpha_i \le \pi/2$ (b) $\alpha_i > \pi/2$

Figure 7: Translation from the sensor coordinator to the robot coordinator.

robot and the measured obstacle becomes less than the boundary value (d_{saft}) of the safety area, which does obstacle collision avoidance, the Avoid becomes activated and the robot's avoidance direction and velocity are determined. At this time, if an obstacle exists in the risk area, the EmergencyStop becomes activated and the robot stops. When an obstacle exists in the safety zone, the cases are divided into Case 1 and Case 2 depending on the positions of the obstacle and robot. Of the values obtained in section 3, when the point near the robot's direction of movement is smaller than the boundary value of the safety zone, it becomes the beginning point ($\theta_{obs,r}$). When the entered value is larger than the boundary value, the previous point becomes the ending point ($\theta_{obs,l}$). When the distance between the robot and the obstacle obtained through circularization becomes the shortest, that distance ($d_{obs,i}$) is called $d_{obs,\min}$ and $\theta_{obs,i}$ at that time is now called $\theta_{obs,\min}$. This can be expressed as (16) in the below.

$$\begin{cases} d_{obs,\min} = \min\{d_{obs,i}\} \\ \theta_{obs,\min} = \{\theta_{obs,i} \mid i : d_{obs,i} = d_{obs,\min}\}. \end{cases} \tag{16}$$

$\Delta\theta$ can be obtained as in (17) using the point where the boundary value meets with the point ($\theta_{obs,\min}$) nearest to the obstacle from the center of the robot.

$$\Delta\theta = \begin{cases} \theta_{obs,l} - \theta_{obs,\min}, & 0 < \theta_{obs,\min} < \dfrac{\pi}{2} \\ \theta_{obs,\min} - \theta_{obs,r}, & \dfrac{\pi}{2} < \theta_{obs,\min} < \pi. \end{cases} \tag{17}$$

- Case 1: When an obstacle exists in the safety zone ($0 < \theta_{obs,\min} < \frac{\pi}{2}$)

As shown in Figure 9, the point where the distance to the obstacle becomes the shortest ($\theta_{obs,\min}$) and the difference between the boundary value and the end point of the obstacle ($\theta_{obs,l}$) ($\Delta\theta$) can be obtained. Then, vertical line D(= $2(d_r + d_{emg})$) that includes the robot radius and the EmergencyStop area is drawn in the normal line direction of the extension line linking the minimum point of the robot center and the obstacle. Using D, $d_{obs,\min}$, angle β is obtained as in (18), and γ , the direction in which the autonomous mobile robot will move, is obtained as in (19).

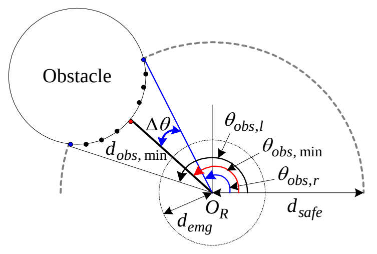


Figure 8: Relation between a robot and an obstacle.

As the direction in which the autonomous mobile robot will move is the direction of γ from the robot's direction of movement, the direction in which the autonomous mobile robot actually moves (ϕ) can be obtained as in (18).

$$\beta = \left| \tan^{-1}\left(\frac{D}{d_{obs,\min}}\right) \right|, \tag{18}$$

$$\gamma = \beta + \Delta\theta. \tag{19}$$

- Case 2: When an obstacle exists in the safety zone ($\frac{\pi}{2} < \theta_{obs,\min} < \pi$)

As shown in Figure 10, when an obstacle exists on the left side of the robot moving direction, the collision avoidance direction can be obtained in the same way as Case 1 with use of (17) – (19). The difference from Case 1 lies in the fact that, as shown in Figure 10, the point where the distance to the obstacle becomes the shortest ($\theta_{obs,\min}$) and the difference between the boundary value and the beginning point of the obstacle ($\theta_{obs,r}$) ($\Delta\theta$) are calculated for application with respect to the information of the two points for the obstacle. As the direction in which the autonomous mobile robot should move is the direction of γ from the robot's direction of movement, the direction in which the autonomous mobile robot actually moves (ϕ) can be obtained as in (20).

$$\phi = \begin{cases} \theta_{obs,\min} + \gamma - \dfrac{\pi}{2}, & 0 < \theta_{obs,\min} < \dfrac{\pi}{2} \\ \theta_{obs,\min} - \gamma - \dfrac{\pi}{2}, & \dfrac{\pi}{2} < \theta_{obs,\min} < \pi. \end{cases} \tag{20}$$

5.3 Determination of velocity and angular velocity for the robot's avoidance of obstacle

To determine the velocity and angular velocity for the robot to avoid an obstacle, the boundary value, the distance to the obstacle, and the robot's maximum velocity and angular velocity were applied using the following expression in the nearness diagram algorithm of Minguez and Montano[11].

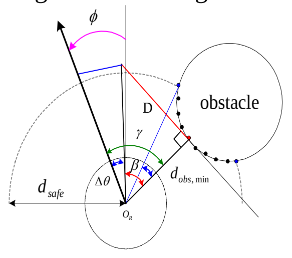


Figure 9: Case 1: concept of robot motion computation($0 < \theta_{obs,\min} < \pi/2$).

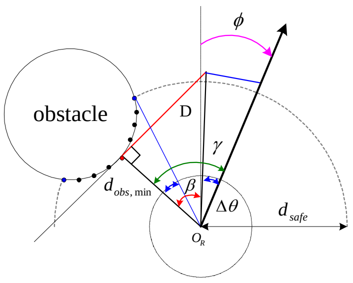


Figure 10: Case 2: concept of robot motion computation($\pi/2 < \theta_{obs,\min} < \pi$).

$$v_o = \frac{d_{obs,\min}}{d_{safe}} \times \left(\frac{\frac{\pi}{2} - |\phi|}{\frac{\pi}{2}} \right) \times v_{\max}, \tag{21}$$

$$\omega_o = \frac{\phi}{\frac{\pi}{2}} \times \omega_{\max}. \tag{22}$$

Here, v_o represents the robot's velocity depending on the approach of the obstacle, d_{obs} the distance between the robot and obstacle, d_{safe} the boundary value to avoid the obstacle, ω_o the robot's angular velocity depending on the obstacle position, and ϕ the direction in which the robot must move.

As shown in (21) and (22), the velocity of the autonomous mobile robot is determined by the direction and distance to the obstacle. The velocity decreases as the obstacle nears the robot, and the angle of the direction of movement becomes larger. The angular velocity is determined by the angle of the direction of movement.

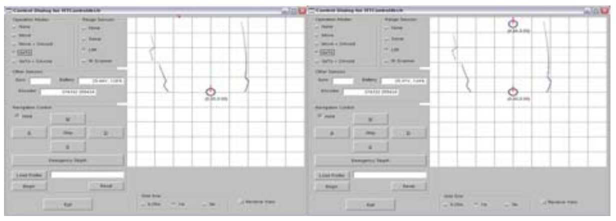
6 Experiment

The performance evaluation of the traveling control architectures proposed in this study is conducted in two ways. First, the evaluation is conducted in the position control mode and the actions of the Goto, Avoid and EmergencyStop are evaluated depending on emergencies like collision with humans, a situation with an obstacle, or a situation without an obstacle when moving toward the target location. Next, the uniform velocity traveling performance is evaluated in the velocity control mode. When the robot proceeds at a certain speed, the precision of traveling can be checked while changing the rotative velocity.

6.1 Position control-based traveling experiment

6.1.1 Goto experiment

This experiment examines whether autonomous traveling takes place successfully in the position control-based action mode (op mode #3: Goto). First of all, a coordinator looks at the environment of the robot described in the GUI environment and clicks on the target point to which the robot will be moved and then the robot moves. In actual autonomous traveling, RTAS delivers the movement information generated from the Process-Supervisor to the Goto and then the robot moves to the target position. Figure 11(a) indicates a picture that gives an order in the GUI environment, and Figure 11(b) shows that the robot traveled precisely according to the given orders. The small error in the position represents the accumulated error due to the encoder. The offset error range set by the program is ± 10 cm.


(a) Goal position setting for Goto test.

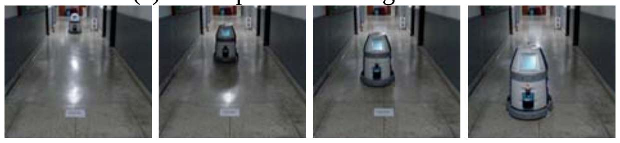

(b) Snap shots

Figure 11: Goto test.

6.1.2 Goto + Avoid Experiment

The next experiment is to examine whether the conversion from the Goto mode (op mode #3) into the Goto + Avoid mode (op mode #4) takes place smoothly when the robot comes across an obstacle on its way to a target position. Figure 12 shows a picture that shows a robot going to a target location while avoiding an obstacle when it encounters a local point. When there is no obstacle, the robot moves toward the target position in op mode #3. But when an obstacle is perceived, the mode in converted into op mode #4 to perform an avoidance motion. When a local point appears, the robot moves toward the position with a larger space from the wall, and when the robot gets out of the obstacle area, it is stopped at the final position by op mode #3.

6.1.3 EmergencyStop Experiment

Figure 13 shows a picture that indicates the results of an experiment. This experiment examines how the robot copes with a collision when it collides with an obstacle on its way to a target point. When the robot runs into a bumper on its way to a target point in op mode #3, the mode changes to op mode #4 and the EmergencyStop gets activated. Then the robot waits until the obstacle passes by. When the obstacle disappears, the robot is stopped at the target position by op mode #3.

6.2 Velocity Control-based Traveling Experiment

6.2.1 Move Experiment

Figure 14 indicates the velocity results generated by the robot when it was given orders to move at the velocity of 10,

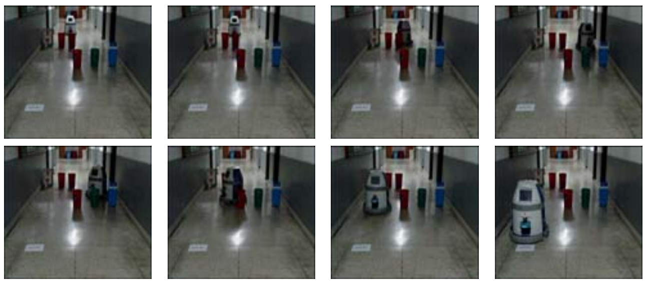


Figure 12: Avoid test.

20, 30 cm/sec for 5 seconds in a stop state. The first second and the last second are the time needed to generate cruise traveling torque irrespective of the velocity. This is due to the specifications offered by the manufacturer. It was confirmed that the robot moved at an exact velocity in the cruise section. As the velocity goes up, the noise gets louder, but it was negligible. When the straight drive velocity is 30 cm/sec, the rotative velocity was changed from between -15 to 15 cm/sec, and the distance traveled was measured through the encoder at an interval of 0.1 sec. The results are shown in the graph of Figure 15. It shows that the robot traveled precisely according to the orders.

6.3 Performance comparison of traveling control architectures

To evaluate the performance of the traveling control architectures proposed above, comparisons with TeamBots[3], and BERRA[10] successfully developed in the past are made. The evaluation method suggested by Orebäck and Christensen[12] was adopted as evaluation riteria. Compared with other control architectures, the traveling control architecture proposed in this study is mounted with a real-time operation system, and it has secured real-time abilities in the processing of sensor information and algorithm computation. In addition, it has a variety of sensors that can be easily attached or detached, and the program size is smaller as well. The proposed traveling control architecture consists of only the traveling part without the whole control architecture of an autonomous mobile robot, but it shows relative excellence in terms of hardware, software and data processing.

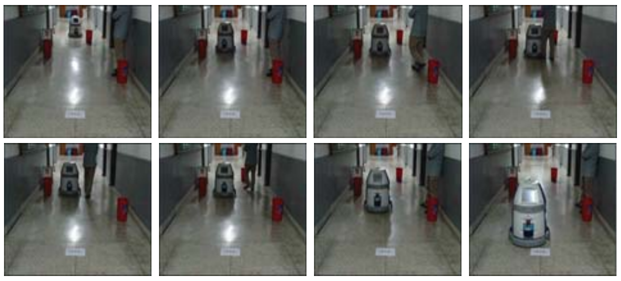


Figure 13: EmergencyStop test.

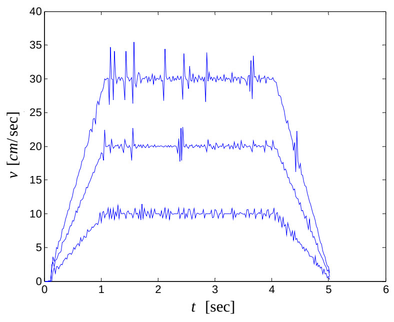


Figure 14: Constant velocity traveling by Move.

7 Conclusion

This study has proposed a control architecture of a reflexive layer for the autonomous traveling of a silver robot under development for the welfare of the elderly. For autonomous traveling, the components forming the reflexive layer in the hybrid deliberative/reflexive architecture were examined and selected from the control architectures appearing in existing studies and reports. Definitions were made for the functions of the components of the Resources, Actions, Action-Coordinator and Motion Controller that form the reflexive layer. Actions were designed as a way to improve mobility, and the mobility and safety were verified through traveling performance experiments. In addition, the application of Kalman Filter led to the minimization of sensor and system errors, and algorithms using geometrical methods have been introduced to ensure simple Avoids with less computation aimed to improve real-time abilities. As a way to create real-time functions, each of the components was designed under the Linux-RTAI environment, which is a real-time operation system. The components designed under the Linux-RTAI environment were applied to an autonomous mobile robot, and autonomous traveling experiments were conducted. As a result, each of the components of the reflexive layer operated in real time, and they moved in a stable manner together with the processor manager, which is a non-real-time component. An experiment also showed that autonomous traveling is actually possible in a hallway. To identify the safety and real-time ability of a robot in the future, it will be necessary to conduct robot experiments that integrate the components of the deliberative layer and sequencing layer as well as the reflexive layer. In addition, various Actions should be additionally designed so that the robot movements can be quicker and softer. Studies on new algorithms will also be necessary so that avoidance of complex obstacles may be possible through the sensor fusion of sonar, IR and others, in addition to LRF.

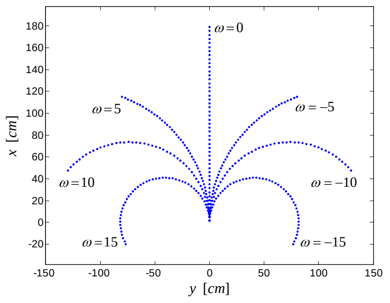


Figure 15: Each of angular velocities at $v = 30$ cm/sec.

Acknowledgment

This research was supported by the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy of Korea.

References

[1] M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino, “Closed loop steering of unicycle-like vehicles via Lyapunov techniques,” IEEE Robotics and Automation Magazine, vol. 2, no. 1, pp. 27-35, 1995.

[2] J. S. Albus, “4D/RCS: A reference model architecture for intelligent unmanned ground vehicles,” Proc. of the SPIE Annual International Symposium on Aerospace/Defence Sensing, Simulation and Controls, Orlando, FL, April 1-5, 2002

[3] T. Balch, “TeamBots”, www.teambots.org

[4] R. A. Brooks, “A robust layered control system for a mobile robot,” IEEE Journal of Robotics and Automation, vol. 2, no. 1, pp. 14-23, 1986.

[5] N. J. Doh, W. K. Chung, Y. Youm, and S. R. Oh, “Shortest path generation scheme for point-stabilization of car-like robot,” International Journal of Applied Electromagnetics and Mechanics, vol. 18(1-3), pp. 115-125, August, 2003.

[6] L. Dozio, and P. Mantegazza, “Linux real time application interface (RTAI) in low cost high performance motion control,” Proc. of the Motion Control 2003, Conference of ANIPLA, 2003.

[7] G. Dudek, and M. Jenkin, Computational Principles of Mobile Robotics, Cambridge University Press, 2000.

[8] G. H. Kim, W. J. Chung, M. S. Kim, and C. W. Lee, “Control architecture design and integration of the autonomous service robot PSR,” Proc. of the 2002 International Conference on Control, Automation, and Systems, Muju, Korea, pp 2379-2384, 2002.

[9] S. J. Kim, and B. K. Kim, “Development of real-time control architecture for autonomous navigation of powered wheelchair,” Journal of Control, Automation, and Systems Engineering(in korean), vol. 10, no. 10, pp. 940-946, 2004.

[10] M. Lindstrom, A. Oreback, and H. I. Christensen, “BERRA: A research architecture for service robots,” Proceedings of the IEEE Conference on Robotics and Automation, San Francisco, CA, USA, pp 3278-3283, 2000.

[11] J. Minguez, and L. Montano, “Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios,” IEEE Trans. on Robotics and Automation, vol. 20, no. 1, pp. 45-59, 2004.

[12] I. Nesnas, A. Wright, M. Bajracharya, R. Simmons, T. Estlin, and W. S. Kim, “CLARAty: An architercture for reusable robotic software,” SPIE Aerospace Conference, Orlando, Florida, March, 2003.

[13] A. Orebäck, and H. I. Christensen, “Evaluation of architecture for mobile robotics,” Autonomous Robots, vol. 14, pp. 33-49, 2003.

[14] P. Ridao, J. Batlle, and M. Carreras, “O2CA2; A new object oriented control architecture for autonomy: The reactive layer,” Control Eng. Practice, vol. 10, pp. 857-873, 2002.

[15] J. Tan, and N. Kyriakopoulos, “Implementation of a tracking Kalman filter on a digital signal processor,” IEEE Tran. on Industrial Electronics, vol. 35, no. 1, pp. 126-134, February, 1988.