An Industrial Manipulator for Shipbuilding: Off-Line Programming and Open Architecture

Ji-Hyoung Lee*, Kyung-Tae Hong*, Seung-Min Oh*, and Keum-Shik Hong*

* School of Mechanical Engineering, Pusan National University; 30 Jangjeon-dong, Gumjeong-gu, Busan 609-735, Korea. (Tel: +82-51-510-2454, Email: kshong@pusan.ac.kr)

Abstract: In this paper, to improve the efficiency of welding and user convenience in the shipbuilding industry, a PC-based off-line programming (OLP) technique and the development of a robot transfer unit are presented. The developed OLP system is capable of not only robot motion simulations but also automatic generations of a series of robot programs. The strength of the developed OLP system lies in its flexibility in handling the changes of the welding robot's target objects. Moreover, for a precise transfer of the robot to a desired location, an auxiliary mobile platform named a robot-origin-transfer-unit (ROTU) was developed. To enhance the cornering capability of the platform in a narrow area, the developed ROTU is equipped with 2 steering wheels and 1 driving wheel. Both the OLP and the ROTU were field-tested and their performances were proven successful.

Keywords: Open architecture, off-line programming, mobile manipulator, virtual reality modeling language, containership building.

1. INTRODUCTION

The off-line programming (OLP) approach is a programming method that utilizes off-line simulation techniques, mimicking the real world, which are set up in advance. The usefulness of OLP is well explained in the literature [1-4]. The advantages of adopting an OLP approach are: i) effective programming of robot-command logics with debugging facilities; ii) easy verification of the validity of a robot program through simulation and visualization; iii) organized documentation with appropriate programs through simulation models; iv) reuse of existing robot programs and easy application to other objects; and v) the cost independence of programming owing to the fact that production can be continued during programming. The developed off-line programs were based on object-oriented programming (OOP) using the virtual reality modeling language (VRML) and various functions including robot simulations.

The purposes of the developed ROTU are first to carry the welding robot to the origin (base position) of a given workcell and then to set the position and orientation of the welding robot correctly according to CAD data. The ROTU has one front wheel and two rear wheels. The front wheel is used for steering purposes and the rear wheels are for both steering and driving. There are numerous studies regarding mobile robots including various applications. Previous studies show that one drive wheel and one steering wheel are enough to steer the mobile robot to a desired position; that is, a mechanism of two wheels may be usable. However, because of the narrow area in a workcell, collisions between the robot body and the surrounding stiffener can occur easily during autonomous navigations. Therefore, motion planning such as that of the sinusoidal input method cannot be applied to welding robots in the shipbuilding industry. In the present study, a three-wheel mobile platform was adopted to increase the steering capability, flexibility, and safety of the robot in narrow areas.

The contributions of this paper are the following. 1) A methodology to apply to a welding robot system for variously shaped objects in hull-assembly lines is suggested; 2) a PC-based OLP using recently issued algorithms, such as the VRML for the simulation environment and computing techniques for the automatic generation of robot programs, was developed; 3) a practical implementation of a mobile robot with two steering wheels using fuzzy reasoning is discussed.

2. PC-BASED OLP

Currently, robot production enterprises provide commercialized software that includes developed robot simulation tools. However, applying this commercialized software to ship construction requires too much time and effort, and therefore utilization of the commercial software is not suitable for shipbuilding purposes. Instead, because of higher expectations for computer systems and the rapid development in graphic interfaces, nowadays, establishing PC-based OLP has become easier and has come to be preferred. Therefore, using OLP for robot systems is suitable for work in the shipbuilding yard, because it is more economical than commercial software that is provided by robot companies.

2.1 System configuration

A typical welding process is composed of a welding robot, a controller, welding equipment, and an on-site industrial computer. Because the size and shape of workpieces in the shipbuilding industry vary greatly, a 6-axis articulated robot is generally used for the purpose of welding. The robot is equipped with a touch sensor to compensate for the difference between CAD data and the actual shape of the workpieces. To increase the path-tracking ability of the robot, an arc sensor is also used.

The OLP system creates job programs based on pre-defined movements and welding macro data. Fig. 1 shows the OLP configuration for grand-assembly. The OLP system is equipped with the following functions: 1) modeling of the shape of a workpiece manually or via the CAD interface; 2) extraction of the weld seam from the shape database; 3) performing of nesting for all weld seams; and 4) generation of job programs for welding robots. The job program consists of a sequence of movement commands for welding robots and welding conditions.

2.2 Robot simulation using VRML

The utilized VRML is a 3D graphic language, which expresses objects and their motions in a space of proper dimensions, and which is useful in constructing a virtual environment on a PC. The salient features of the VRML are as follows: first, it is easier to interpret because the text is expressed in a grammatical structure of functions; second, one can easily obtain VRML models from other CAD software, because converting a CAD drawing to a VRML model is

possible in most 3D CAD software; third, the VRML model contains vertex data, which makes it easy to extract useful specific point information; fourth, because the 3D objects are modeled by VRML, which can be shown on the Internet, OLP using the Internet is an option.

Fig. 2 illustrates the structure of the developed PC-based OLP system. The user screen consists of four fields for user-friendly interface: the set-up menus, the main graphic simulation screen, a jog panel for off-line teaching, and a message window. The menu bar consists of a CAD interface, a block arranging algorithm, a path planning algorithm, and an automatic-robot-program generator.

To render a robot, a robot initialization file is used. The robot initialization file contains the data of the robot body modeling, the link parameters, the limit values of individual joints, and the home position data. In addition, the user can arbitrarily specify the robot base position, so that the initial position of the robot system can be easily set. Also, through the manipulation of kinematics data, the base coordinate frame can be easily placed at a desired position. Hence, the reachability of the end-tool and possible collisions with surrounding parts can be easily examined through simulations in a virtual environment.

The results of simulated motions will approach to those of the real ones, if the algorithms in the simulation program including kinematics, robot motion planning, and the robot language interpreter are identical to those of the real ones. Because the sampling time of the control input is 16 msec whereas the interpolation time of a robot motion is 5 msec, the robot motion is updated every 16 msec in simulations. Also, multi-threads called for by a 16 msec timer are used for multi-robot simulation. In this case, one thread can initiate some of the functions shared with other threads, such as the robot language interpret function, the motion planning function, and the starting command function, at the same time, which results in memory leakage or malfunction. So the multi-threads and the *CCriticalSection* of VC++ work together.

To implement various 3D solid models and motions on a PC, a structured graphical representation of the nodes, see Fig. 3, is acquired. Each link can be replaced with a newly designed link without influencing other graphic objects. The 3D solid model of each link is defined as *m_Arm*[*n*], whereas each link's motion engine is defined as *myRotor*[*n*], where *n* represents the *n*-th link. The term *m_Arm*[*n*] is a variable name that stores the n-th link model, and the term *myRotor*[*n*] is the variable name of the associated motion engine. The basic configuration is a parallel combination of the motion engine and the link model of each link. Accordingly, the motion of the $(i+n)$th link, where $n=1,2,3,\ldots$, is affected by the movement of the i-th link. Auxiliary graphic objects such as axes, texts, and welding lines are added to the top-node defined as *m_pSceneRoot* directly, in order to be independent of the robot's movements. For example, to display the axis of the teaching points and the welding line, *m_pAxisSep* and *m_pLine* are attached to the *m_pSceneRoot* node independently of the motion of the related nodes *m_pLoadBlockSep* and *GantrySep*, as shown. The axis graphic node is added to the *m_pSceneRoot* node whenever the user introduces a new teaching point. The added axis graphic node is counted, and the entire axis in the simulation window has its own number. By clicking the axis in the simulation window, the simulation window displays the data of the selected teaching point. In the same way, the *m_pLine* node containing the line graphic object is added to the *m_pSceneRoot* node.

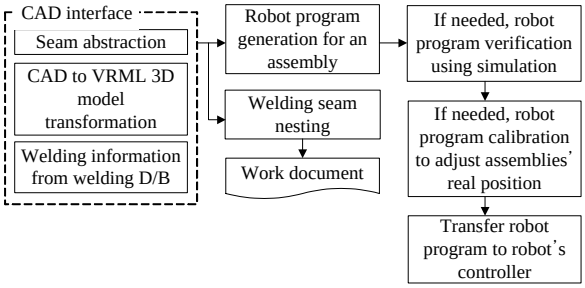


Fig.1. Flow chart of the OLP for a grand-assembly.

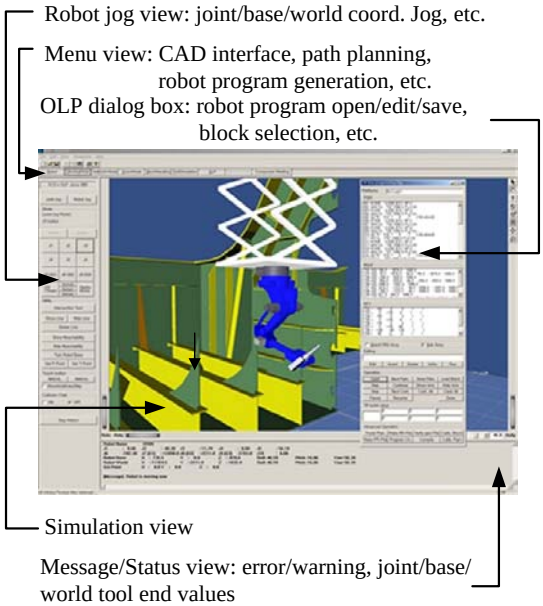


Fig. 2. The structure of the developed PC-based OLP.

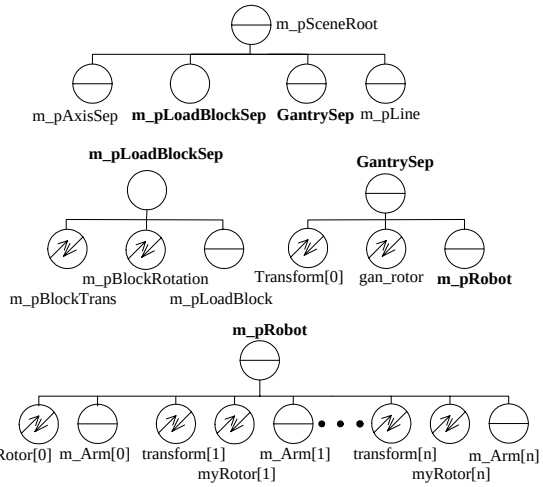


Fig. 3. Simulation environment: graphic nodes.

2.3 Automatic generation of robot programs

In operating welding robots in a shipyard, the largest portion of time consumption lies in robot programming. Particularly, for workpieces of different shapes and sizes, more time is required to render the robot programs operable in real-time. To minimize time consumption, robot programs are often generated automatically from the welding information obtained from CAD data. First, according to the analysis of

the shape of the workpiece, the shape can be represented in simple geometry such as that of a scallop, a hole, a horizontal line, a horizontal curve and a vertical line, and others. The programs for these simple geometries are pre-created and saved in the robot program D/B. When the workpiece is allocated to the robot system, the robot program generation algorithm divides the workpiece into a number of simple, pre-defined geometries. Next, the robot programs required for the respective simple geometries are selected from the D/B and combined together to complete the entire program for a given workpiece.

Fig. 4 is a flow chart representing automatic robot program generation. The robot program generation algorithm is composed of 5 sub-routines: input data conversion, via-point creation, robot program selection using simple geometries, compilation of the combined robot programs of simple geometry, and robot program writing. The input data conversion routine creates teaching points for each seam of the work place by the methodology explained above. In the via-point creation routine, all of the via-points are calculated in such a way that no collision occurs in moving from one teaching point to another teaching point. The collision-free path is obtained by pre-simulation results that are obtained for all of the shapes of the workpieces. In the robot program selection routine, the simple geometries of a workpiece are matched to the respective robot programs. The writing program routine rewrites the robot program to fit it to the format of the robot language. Moreover, it sorts the teaching points according to number and matches each teaching point in the robot program to the respective position.

2.4 Robot base positioning

Determining the robot base position of a workcell via OLP is important. The primary aspects of the robot base in a grand-assembly are: 1) the robot base is located on the center between the left-bottom edge and the right-bottom edge; 2) the distance from a transversal stiffener is 700 mm; and 3) the robot base is parallel to the workpiece plate. However, the determination of the robot base of a workcell varies according to the shape of a workpiece. At this time, an OLP simulation is applied to decide whether the workpiece is weldable or not, and to modify the robot base position. After the robot base position is determined, the robot program obtained from the automatic robot program generator is calibrated to a job program according to the modified robot base. Finally, the modified robot base position is sent to the reference position of the ROTU and the modified job program is downloaded to the robot's controller.

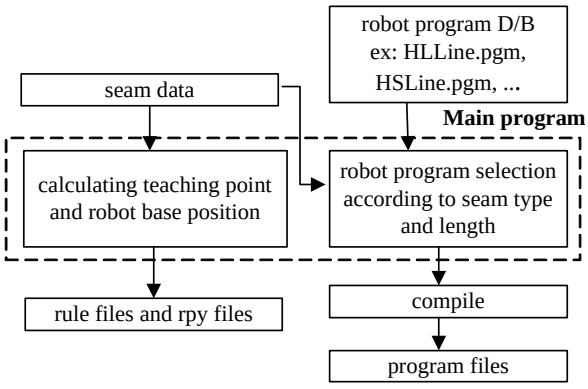


Fig. 4. Automatic program generation.

Table 1. Servo motor specifications.

	Diameter	Max. velocity	Torque
Rear wheel driving	92 mm	52.1 m/min	9.8 Nm
Rear wheel steering	92 mm	62 rpm	11.5 Nm
Front wheel steering	107 mm	53 rpm	8.5 Nm

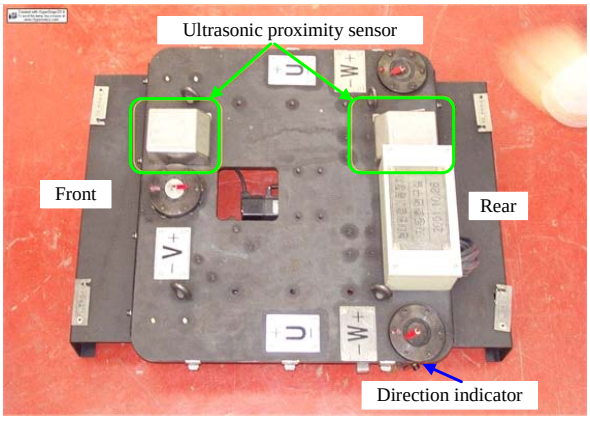


Fig. 5. Upper view of the ROTU.

3. ROBOT ORIGIN TRANSFER UNIT

3.1 System description

Table 1 summarizes the specifications of the servo motors used in the front and rear wheels. Other detailed information of the ROTU are: weight 72.5 kg, max pulling force 549.9 N, static pulling force 183. 3 N, allowed load 200 kg, length 774 mm, width 586 mm, and height 209 mm. Fig. 5 and Fig. 6 show the upper and lower views, respectively, of the developed ROTU.

Fig. 7 is a schematic of the ROTU for the purpose of analysis. Because the frame is rigid, the component of V_1 in the y' axis must be equal to the component of V_2 in the y' axis. That is,

$$V_1 \cos\alpha_1 - V_2 \cos\alpha_2 = 0\,, \tag{1}$$

where α_1 and α_2 are the front- and rear-wheel steering angles, respectively. Because the two perpendicular axes of the moving directions of the front and rear wheels intersect at an instantaneous center of rotation, the angular velocity of the ROTU is obtained as

$$\dot{\varphi} = \frac{d\varphi}{dt} = \frac{V_1 \sin\alpha_1 - V_2 \sin\alpha_2}{a}\,, \tag{2}$$

where φ is the angular displacement of the robot body and a is the body length. Let the components of the velocity of the body center in the x' and y' directions be V_x and V_y, respectively. Then, they are

$$V_x = -V_2 \sin\alpha_2 - c\dot{\varphi}\,, \tag{3}$$

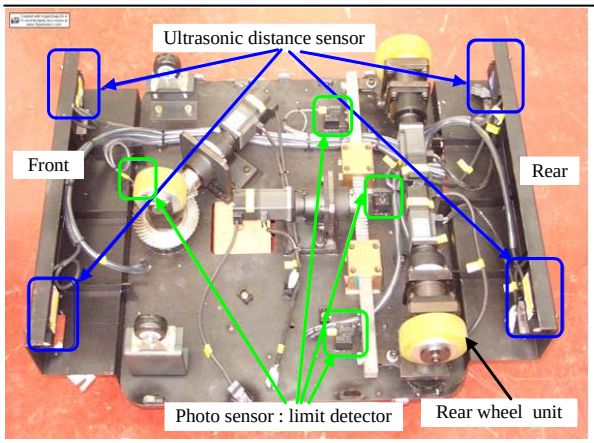


Fig. 6. Lower view of the ROTU.

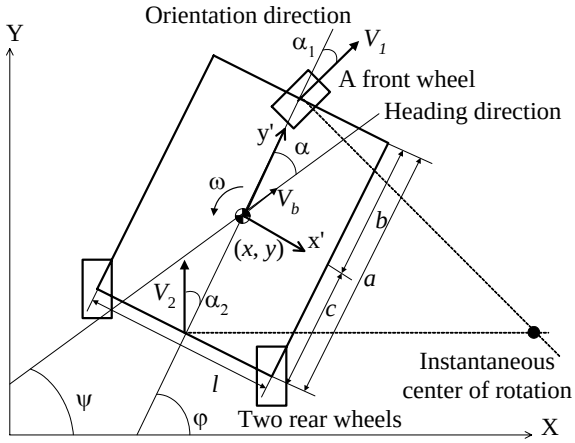


Fig. 7. A schematic of the ROTU.

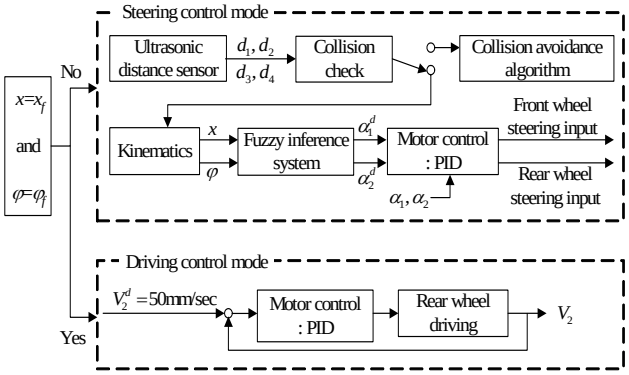


Fig. 8. Block diagram of the two-mode control scheme

$$V_y = V_2 \cos\alpha_2 , \tag{4}$$

where c is the distance between the body center and the driving wheel. Therefore, the velocity of the body is calculated, using (3) and (4), as

$$V_b = \sqrt{V_x^2 + V_y^2}$$

$$= V_2 \sqrt{\cos^2\alpha_2 + \left(\frac{c\dot{\varphi} + V_2 \sin\alpha_2}{V_2} \right)^2} , \tag{5}$$

And the heading directional angle is defined as

$$\psi = \varphi - \alpha , \tag{6}$$

where $\alpha = \tan^{-1}\left(V_x / V_y\right) = -\tan^{-1}\left(\frac{V_2 \sin\alpha_2 + c\dot{\varphi}}{V_2 \cos\alpha_2} \right)$.

If we define $\alpha' = -\alpha$, then, (6) can be rewritten as

$$\psi = \varphi + \alpha' , \tag{7}$$

where $\alpha' = \tan^{-1}\left(\frac{V_2 \sin\alpha_2 + c\dot{\varphi}}{V_2 \cos\alpha_2} \right)$. Finally, the absolute velocity of the ROTU is

$$\begin{aligned} \dot{x} &= V_b \cos\psi , \\ \dot{y} &= V_b \sin\psi . \end{aligned} \tag{8}$$

Here, let the state vector be $z = [x, y, \varphi]^T$; then, the state equation is obtained as

$$\dot{z} = g\left(V_2,\ \alpha_1, \alpha_2, \dot{\varphi}\right) , \tag{9}$$

where

$$g(\cdot) = \begin{bmatrix} V_2 \sqrt{\cos^2\alpha_2 + \left(\frac{c\dot{\varphi} + V_2 \sin\alpha_2}{V_2} \right)^2} \cos\psi \\ V_2 \sqrt{\cos^2\alpha_2 + \left(\frac{c\dot{\varphi} + V_2 \sin\alpha_2}{V_2} \right)^2} \sin\psi \\ \frac{V_1 \sin\alpha_1 - V_2 \sin\alpha_2}{a} \end{bmatrix} .$$

3.2 Steering control using fuzzy reasoning

In this paper, a novel two-mode control scheme is proposed, as depicted in Fig. 8. The first mode is the steering control mode that steers x and φ to their desired values. The second mode is the driving control mode, in which the y-directional movement is controlled using the rear wheel driving motor only.

The migration of the ROTU to its designated position defined via the OLP is done after the welding robot is lowered to the ROTU. Fig. 9 shows a top view of the ROTU commissioned in a workcell.

The following variables are introduced: x_i is the initial X-directional position; y_i is the initial Y-directional position; φ_i is the initial rotational angle; x_f is the target X-directional destination position; y_f is the target Y-directional destination position; and φ_f is the target rotational angle. Then, the control problem is defined as a problem of steering the ROTU from $\{x_i, y_i, \varphi_i\}$ to

$\{x_f, y_f, \varphi_f\}$. The welding robot's base position is defined as {O}, as shown in Fig. 9, and then, the destination position of the ROTU is $x_f = 0$, $y_f = 0$, and $\varphi_f = 0$ in the reference frame {O}.

3.2.1 The steering control

As shown in Fig. 9, the controller measures the distance between the ROTU and the longitudinal stiffener using four ultrasonic distance sensors. And then the initial position, x_i, is obtained as

$$x_i = \frac{1}{2}\{(d_1 + d_2) - (d_3 + d_4)\}, \quad (10)$$

When the system is used in an open-type workcell, in which the left or right longitudinal stiffener does not exist, the equation

$$x_i = \frac{1}{2}\{W - H - (d_1 + d_2)\}, \quad (11)$$

is used, where W is the width of the workcell and H is the width of the ROTU. Also, the rotational angle φ_i of the ROTU is defined as

$$\varphi_i = \tan^{-1}\left(\frac{d_1 - d_2}{a}\right), \quad (12)$$

where d_1, d_2, d_3, and d_4 are the distances measured by four ultrasonic distance sensors. Then, it is necessary to check whether the ROTU is inside or outside the collision area, determined by using the ultrasonic proximity sensor signals, d_5 and d_6. If the ROTU enters a collision area, the collision avoidance algorithm can be executed. Otherwise, the steering control for $\{x, \varphi\}$ initiates. While the ROTU is working, the controller monitors d_5 and d_6 to prevent a possible collision.

To simplify the steering problem, the driving velocity, V_2, is fixed as ±50 mm/sec. And if we define a reduced-order state variable as $z_s = [x, \varphi]^T$, then the control problem of (9) is simplified as

$$\dot{z}_s = g_s(\alpha_1, \alpha_2, \dot{\varphi}), \quad (13)$$

where $g_s(\cdot) = \begin{bmatrix} V_2\sqrt{\cos^2\alpha_2 + \left(\frac{c\dot{\varphi} + V_2\sin\alpha_2}{V_2}\right)^2}\cos\psi \\ \frac{V_1\sin\alpha_1 - V_2\sin\alpha_2}{a} \end{bmatrix}$.

In (13), the control inputs are the front wheel steering angle, α_1, and the rear wheel steering angle, α_2. Both of the steering angles are calculated by fuzzy reasoning. The two inputs of this fuzzy reasoning are x and φ, the current pose of the ROTU. The fuzzy rule is based on the geometric relation of the steering angles and the body rotation direction. For positive α_1 and α_2, if $\alpha_1 > \alpha_2$, then the ROTU rotates in the clockwise direction, and if $\alpha_1 < \alpha_2$, then the ROTU rotates in the counterclockwise direction. The fuzzy membership function is defined as a triangular type. The derived fuzzy rules are shown in Table 3. In a defuzzification algorithm, the center of mass method is applied to the fuzzy reasoning. At each sampling time, the current position and steering angle are calculated. The obtained steering angle is used as a reference value in steering.

3.2.2 The driving control

With the setting of all of the steering wheels to 0, the ROTU moves only forward and backward until the ultrasonic approximate sensor sends a signal to the controller. When the ultrasonic approximate sensor is tuned to detect the constant distance, the signal of the approximate sensor indicates the Y-directional destination position.

Table 3. Fuzzy rules for adjusting steering angles (NB = negative big; NS = negative small; ZO = zero; PS = positive small; and PB = positive big).

(a) Front-wheel steering angle.

x \ φ	NB	NS	ZO	PS	PB
NB	PB	PB	PB	PS	ZO
NS	PB	PB	PB	PS	ZO
ZO	PB	PS	ZO	NS	NB
PS	ZO	NS	NB	NB	NB
PB	ZO	NS	NB	NB	NB

(b) Rear-wheel steering angle.

x \ φ	NB	NS	ZO	PS	PB
NB	ZO	PS	PB	PB	PB
NS	ZO	PS	PB	PB	PB
ZO	NB	NS	ZO	PS	PB
PS	NB	NB	NB	NS	ZO
PB	NB	NB	NB	NS	ZO

Table 4. Comparison of robot setting times.

	The developed ROTU	Maker A's ROTU	Human operator
Setting time	16 sec	21 sec	30 sec

ROTU: robot origin transfer unit

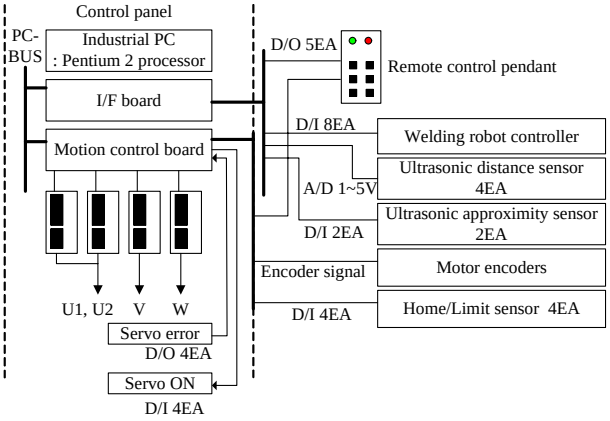
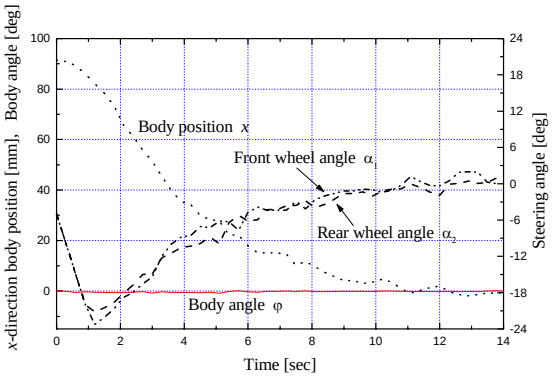
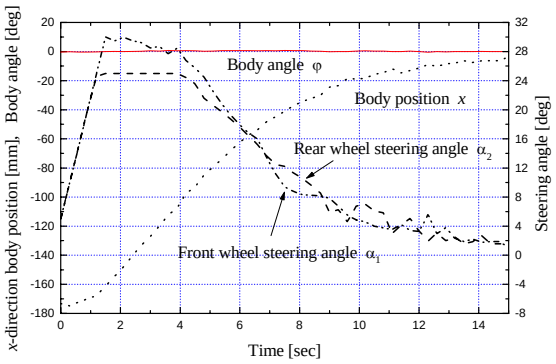


Fig. 9. Hardware architecture.



(a) x_i = 90 mm and φ_i = 0°.



(b) x_i = 180 mm and φ_i = 0°.

Fig. 10. Experimental results.

4. EXPERIMENTAL RESULTS

4.1 Implementation

The tools utilized in OLP are as follows. The PC O/S of a Pentium IV 2.4 GHz processor with Windows 2000 was used for computation, and Visual C++ was used for programming. Open Inventor was used to implement the graphic environment for the OLP. Because Open Inventor is a graphic library that provides collision detection algorithms, it is useful in constructing graphic environments. To increase the efficiency of the graphic processing ability, the selection of a video card is very important. Hence, the implemented video card has a 64 MB frame buffer memory and a 128 MB texture memory. Also, the video card was optimized to process Open GL. On account of OOP, the user can selectively use each function of OLP, which automatically performs all functions in real time. Considering users' convenience, for the grand- and mid-assembly welding robot systems, robot simulation and robot program automatic generation functions were mainly used.

The ROTU is composed of a control panel, a mobile robot, and a remote control box. The controller employs the Intel Pentium MMX 233 MHz, 80Mb solid-state disk, which prevents vibration and dust, a commercial 4-axis motion controller, a signal I/O board, and the Windows NT Embedded O/S. The encoder used in the controller is an incremental type, and the resolution is 4,000 pulse/rev. Because the Windows NT Embedded O/S allows the user to select the optimized component in implementing a controller, the booting time and the O/S size can be reduced significantly.

4.1 Experimentation

The experimental results are shown in Fig. 10. The sampling time was 300 msec. The experiments were performed for the cases of the ROTU's initial position and rotation offset value that occurs frequently while a robot operator unloads a welding robot in a workcell. Table 4 compares the setting times of the developed ROTU and Maker A's ROTU with that of a human operator.

5. CONCLUSIONS

To improve the user convenience and efficiency of the welding robots in the shipbuilding industry, a PC-based off-line programming methodology and a robot transferring unit, named ROTU, were developed. The developed OLP system provides both robot simulations and automatic robot program generation. Because the graphics were made in a VRML environment, the developed OLP is highly compatible with other software, which allows the use of OLP through the Internet. Also, the mobile-robot-type ROTU is very efficient in shortening the setting time of the welding robot in various-size-and-shape workcells. The quality and stability of the suggested algorithms were verified through experiments. In short, OLP helps operators to access the robot system easily and the ROTU helps them to reduce their physical efforts. The methodology outlined in this paper is practical and can easily be applied to other areas.

ACKNOWLEDGMENT

This work was supported by the Research Center for Logistics Information Technology, Pusan National University, funded by the Ministry of Education and Human Resources Development, Korea.

REFERENCES

[1] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, 1986.

[2] C. S. Kim, K. S. Hong, and Y. S. Han, "PC-based off-line programming in shipbuilding industry: open architecture," *Advanced Robotics*, vol. 19, no. 1, 2005.

[3] N. Kobayashi, S. Ogawa, and N. Koibe, "Off-line teaching system of a robot cell for steel pipe processing," *Advanced Robotics*, vol. 12, no. 3, pp. 327-332, 2001.

[4] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, 1994.

[5] Y. Nagao, H. Urabe, F. Honda, and J. Kawabata, "Development of a panel welding robot system for subassembly in shipbuilding utilizing a two-dimensional CAD system," *Advanced Robotics*, vol. 14, no. 5, pp. 333-336, 2000.

[6] J. Wernecke, *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor*, Open Inventor Architecture Group, 1994.

[7] J. M. Yang and J. H. Kim, "Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robot," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 578-587, 1999.