Advanced Robotics, Vol. 19, No. 4, pp. 435–458 (2005)

Also available online - www.vsppub.com

Full paper

PC-based off-line programming in the shipbuilding industry: open architecture

CHANG-SEI KIM [1], KEUM-SHIK HONG [3,*] and YONG-SEOP HAN [2]

[1] *Robot Research and Development Team, Daewoo Shipbuilding and Marine Engineering Ltd., 1 Aju-dong, Geoje-si, Gyeongsangnam-do 656-714, South Korea*
[2] *Industrial Application Research and Development Institute, Daewoo Shipbuilding and Marine Engineering Ltd., 1 Aju-dong, Geoje-si, Gyeongsangnam-do 656-714, South Korea*
[3] *School of Mechanical Engineering, Pusan National University, San 30, Jangjeon-dong, Gumjung-gu, Busan 609-735, South Korea*

Received 25 June 2004; accepted 14 October 2004

Abstract—In this paper, a PC-based off-line programming (OLP) method using the virtual reality modeling language is proposed. The developed OLP system consists of simulation programs, a block-arrangement algorithm, an optimal traveling-path generation algorithm, an automatic robot program generator and the TRIBON CAD interface. The strength of the developed PC-based OLP system lies in its flexibility in handling the changes in the robot's target objects. The operator can generate robot programs very easily and quickly. Possible applications of the developed OLP can be extended to port automation, container loading/unloading processes as well as painting and grinding processes in the shipbuilding industry.

Keywords: PC-based control; open architecture; off-line programming; crane and shipbuilding; industrial robot; virtual reality modeling; genetic algorithm.

1. INTRODUCTION

The shipbuilding industry is steadily advancing by introducing robots to its work fields so as to increase productivity and improve working conditions [1–3]. In particular, robot applications have yielded a large productivity improvement in hull assembly welding and have reduced work-related musculosketetal disorders of workers. However, because the shapes and sizes of workpieces vary, and, furthermore, because the operating environments of robots are not stable, it is difficult to operate robots in welding processes, particularly in subassembly lines. Moreover, in order to achieve a desired production schedule, the ability to generate robot programs in real-time for various applications is of great importance. Currently the

*To whom correspondence should be addressed. E-mail: kshong@pusan.ac.kr

operation of industrial robots is through either on-line teaching or off-line programming (OLP) [4–11].

On-line teaching is, by definition, a technique of generating robot programs using a real robot system, whereas OLP is a method using simulations that are set up in advance. On-line teaching may be suitable for jobs for which a robot only needs to repeat a monotonous motion using one pre-written program that applies to identical sizes or objects. However, in work places where objects are constantly changing, on-line teaching will cause problems due not only to the decrease of productivity caused by halting the robots while reprogramming, but, more importantly, through not being able to revise work errors that on-line programming itself can cause. Hence, the more profitable method of building a work program is using OLP, while only applying programs that were already verified to be effective for the job.

The advantages of using OLP are: (i) effective programming of robot-command logic with debugging facilities, (ii) easy verification of the validity of the robot programs through simulation and visualization, (iii) organized documentation through simulation models with appropriate programs, (iv) reuse of existing robot programs and easy application to other objects and (v) cost independence of production due to the fact that production can be continued while programming.

Currently, research on robotic system simulation is prevalent [12–16], and robot production enterprises provide commercialized software for robot simulations such as ROBCAD and IGRIP, which include developed simulation tools for the robots. However, applying this commercialized software to ship construction requires extra preparation, including users becoming fluent with CAD systems, complete modeling of the work object and development of language translators that will work with robot manufacturing companies. In short, because it takes too much time and effort, the utilization of commercial software for robot systems is not suitable. Instead, because of high expectations for computer systems and the rapid development in graphic interfaces, nowadays establishing a PC-based simulation environment has become easier and has come to be preferred. Therefore, using OLP for robot systems is suitable for work in a shipbuilding yard because it is more economical than commercial software that is provided by robot companies.

In this paper, the results of our work in the development of PC-based off-line programming for welding robots on hull assembly lines are presented. The programs were developed based on object-oriented programming (OOP). Using the virtual reality modeling language (VRML), various functions including robot simulations, block placement simulations, optimal trajectory generations and automatic CAD interfaces were performed. The VRML utilized is a three-dimensional (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: (i) it is easier to interpret because the text is expressed with a grammatical structure of functions, (ii) 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, (iii) the VRML model

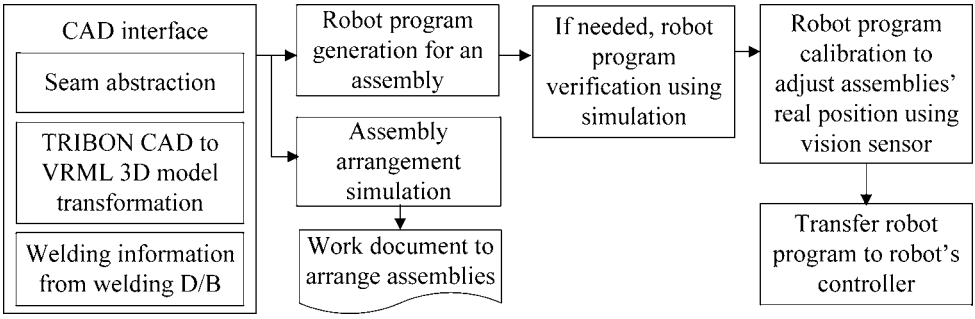


Figure 1. Flowchart of the OLP and graphic simulation.

contains vertex data which makes it easy to extract useful specific point information and (iv) because the 3D objects are modeled by VRML, which can be shown on the Internet, OLP using the Internet is an option.

Figure 1 shows a flowchart of the developed OLP. If the shape of the workpiece changes, then a new program for the new shape is generated by simulation. First, if the workpiece is identical to an old one, then an appropriate program is automatically loaded from the database (D/B). Simulations using OLP are particularly effective when creating a robot program for movements on a critical surface, whereas an automatic program is used in reference to objects with predetermined surfaces. Second, depending on the work schedule, OLP provides a work-order document of a workplace arrangement for the blocks. Third, after arranging the blocks according to the document, a vision sensor verifies and revises the positions of the blocks. Finally, the off-line program sends the robot program to the controller using TCP/IP communication.

The contributions of this paper are as follows: (i) a methodology for applying a welding robot system that works for variously shaped objects, especially for assembly lines of shipbuilding, is suggested, (ii) the functions required to implement OLP successfully and the development of PC-based OLP that helps on-site operators handle a robot system easily are explained and (iii) the practical implementation of recently issued algorithms such as the VRML of the simulation environment, the geometrical computation of the CAD interface, the computing techniques of the automatic generation of robot programs and the genetic algorithm (GA) of robot path planning are shown.

The paper is structured as follows. In Section 2, the application of OLP in a welding robot system is explained. In Section 3, the general structure of the robot simulation program is discussed. Section 4 describes the algorithm for the application of the CAD interface and the robot program to the actual welding plant. In Section 5, the process of finding the optimum trajectory for scattering the blocks based on a block arrangement simulation is explained. Section 6 presents an example of an application. Finally, Section 7 provides conclusions.

2. SYSTEM CONFIGURATION

Figure 2 shows the welding robot system that is being used at Daewoo Shipbuilding and Marine Engineering (South Korea). The system is composed of a welding robot, a controller, a gantry crane, welding equipment and an on-site industrial computer. The same configuration is being used for two types of welding processes: grand- and mid-assembly. A four-axis gantry crane and a hanging-type six-axis robot to increase the moving ability of the robot system are used for the subassembly. Because the size and shape of workpieces in the shipbuilding industry vary greatly, a six-axis articulated robot is generally used for welding. The robot controller consists of a Pentium II processor, three motor interface boards and a digital signal processing board. Because one motor interface board controls four motors, room for adding more control boards for auxiliary actuators has been reserved. The QNX is used as a real-time operating system. The robot is equipped with a touch sensor to compensate for the difference between CAD data and the actual shape of workpieces. To increase the robot's path-tracking ability, an arc sensor is also being used.

Figure 3 shows an example of the robot program, named the standard program, which consists of three separate parts: a program file, an rpy file and a rule file. Rpy represents the roll, pitch and yaw values of the orientation of the tool. The program file describes the sequence of robot motions that is written in robot language, the rule file contains the tool position values of the teaching points indicated by the

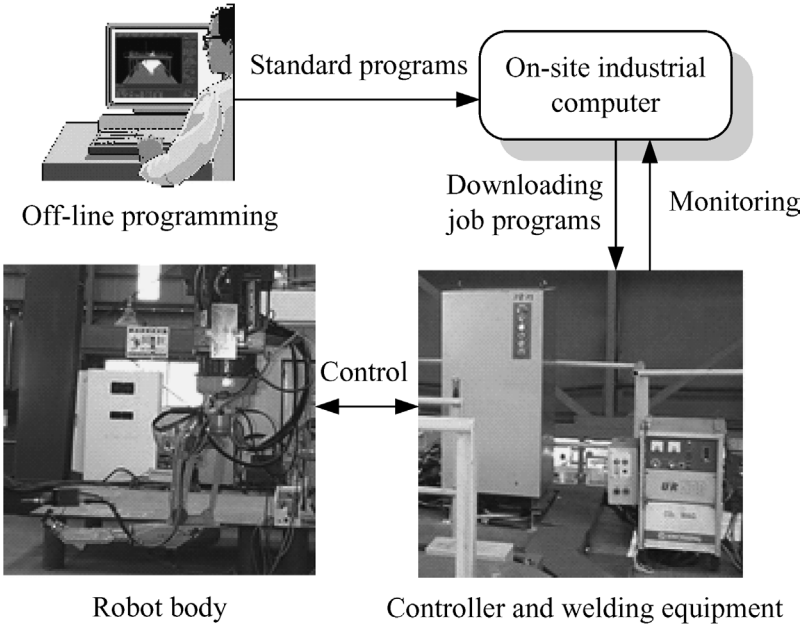


Figure 2. Configuration of the welding robot system.

```
:FR65-74sl.pgm
  001 RHOME  V=030.0(%) HF=1
  002 GETP    P01 0     0     0     CRD=BASE
  003 GMOVJ  T01 V=030.0(%) PL=0
  004 RHOME  V=030.0(%) HF=2
  005 RMOVJ  T02 V=030.0(%) PL=0
  006 GMOVJ  T03 V=030.0(%) PL=0
  007 GMOVJ   T04 V=030.0(%) PL=0
  008 GETP     P02 0     0     0     CRD=BASE
  009 RMOVL   T05 V080.0 PL0 D0
  010 RTOUCH V=035.0 X1  Y0  Z0  L100 P00
  011 RIMOV    V=030.0 x=-10  y=0    z=0    CRD=BASE
  012 RTOUCH V=035.0 X0  Y0  Z-1 L100 P00
```

```
: FR65-74sl.rpy
  T02 =   180     0     0     1     1
  T03 =  -180   -47   180     1     1
  T04 =   180     0     0     1     1
  T05 =    90     0     0     1     1
  T06 =   180   -47    90     1     1
```

```
: FR65-74sl.rule
  T02 G52 2064.0   -2389.0   1680.0
  T03 A52 2734.0   -2389.0   44.0   2064.0   -2389.0   1680.0   0.0
  T04 A52 2734.0   -1889.0   44.0   2064.0   -1889.0   1680.0   0.0
  T05 A52 2734.0   -2889.0   44.0   2064.0   -2889.0   1680.0   0.0
  T06 G52 2177.6   -2344.2   1680.0
```

Figure 3. An example of the standard program consisting of three parts: a program file (top), a rpy file (middle) and a rule file (bottom).

program file and the rpy file contains the orientation values of the teaching points. In order to apply variously sized, but identically shaped, target objects without modifying the pre-generated robot program, the teaching points in the standard program are written in a variable form.

Before the controller executes the standard program, information regarding the real size of the workpiece and the welding conditions should be contained in the standard program. The standard program including this information is referred to as a job program. The size information is obtained by the CAD interface and the welding conditions are gathered from a database in which all previous experimental data have been stored. TCP/IP transfers the job program to a robot controller. When the controller receives the job program and the start signal, the controller interprets the job program and controls the movements of the robot.

Figure 4 depicts the defined coordinate systems: {W} denotes the fixed world (reference) coordinate system attached to the ground, {B} denotes the base coordinate system affixed to the base of the robot, whose position will be determined by the movement of the gantry, {O} refers to the object coordinate system attached to the workpiece and {T} represents the coordinate system attached to the tool center.

3. ROBOT SIMULATION

Figure 5 illustrates the structure of the developed PC-based OLP system. The user screen consists of four fields for a 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

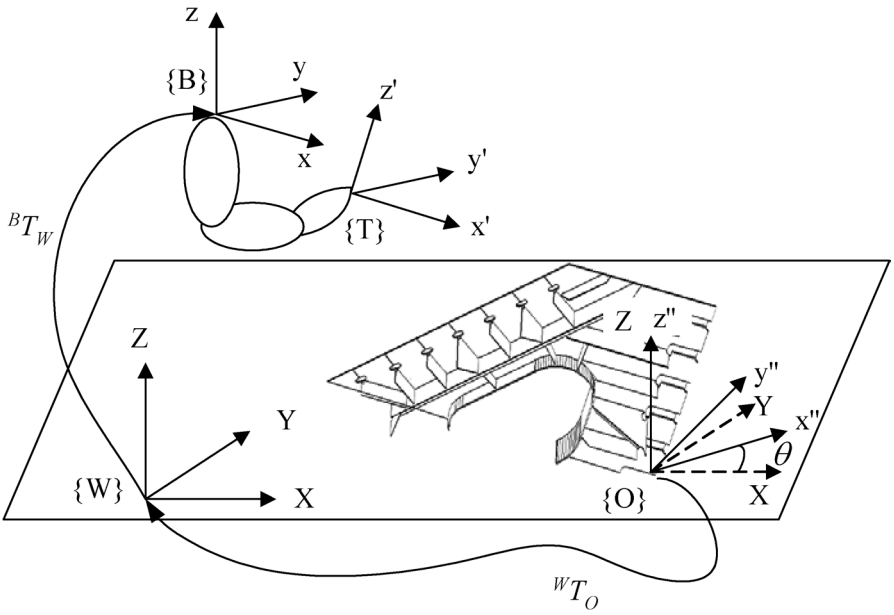


Figure 4. The coordinate systems defined: world, base, object and tool.

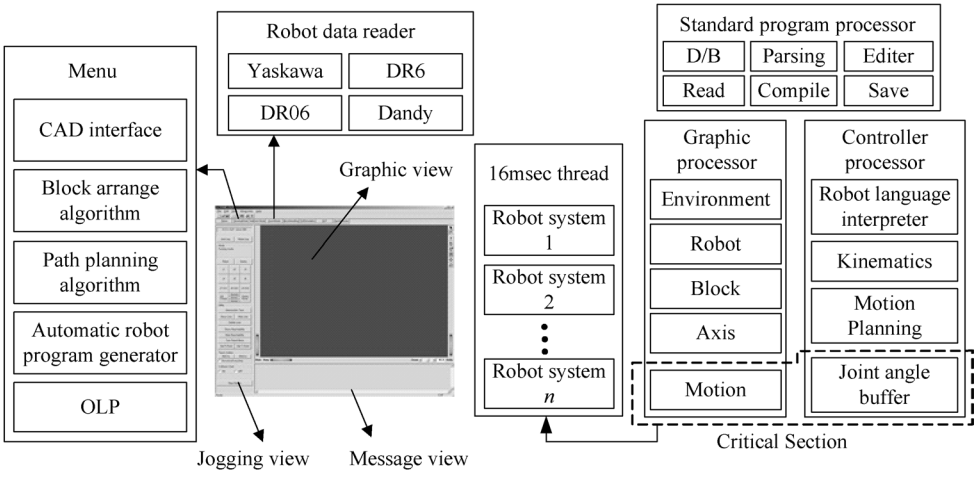


Figure 5. The structure of the PC-based OLP developed.

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-effector and possible collisions with the surrounding parts can be easily examined through the simulations in a virtual environment.

Two types of simulation modes are provided: a teaching mode and an execution mode. Robot teaching tells the robot what to do. Because the operator can easily move the robot in various motions with *via*-points using the teaching mode, this mode is very helpful for operators. The teaching mode includes two jog functions: a joint jog function that moves the joint actuators in relation to the joint coordinates and a coordinate jog function that moves the robot according to a given coordinate frame, as shown in Fig. 4. In the program execution mode, all of the robot motions written to a standard program are automatically, simultaneously and continuously executed.

Two simulations are illustrated: Fig. 6 shows the simulation of a grand-assembly, while Fig. 7 shows the simulation of a subassembly for which the robot is the hanging type. Figure 8 is a picture of an actual grand-assembly.

The simulated motions approach the real ones, because the algorithms of the simulation program including kinematics, the robot motion planning and the robot language interpreter are identical to those of the real controller's. Because the control input sampling time is 16 ms, whereas the interpolation time of a robot motion is 5 ms, the robot motion is updated every 16 ms in simulations. Also,

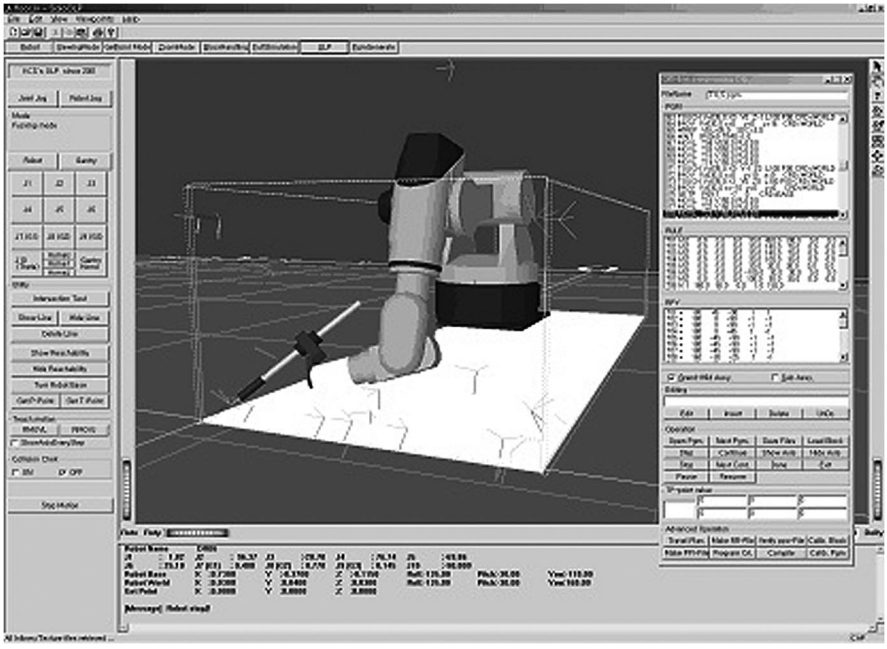


Figure 6. A grand-assembly simulation.

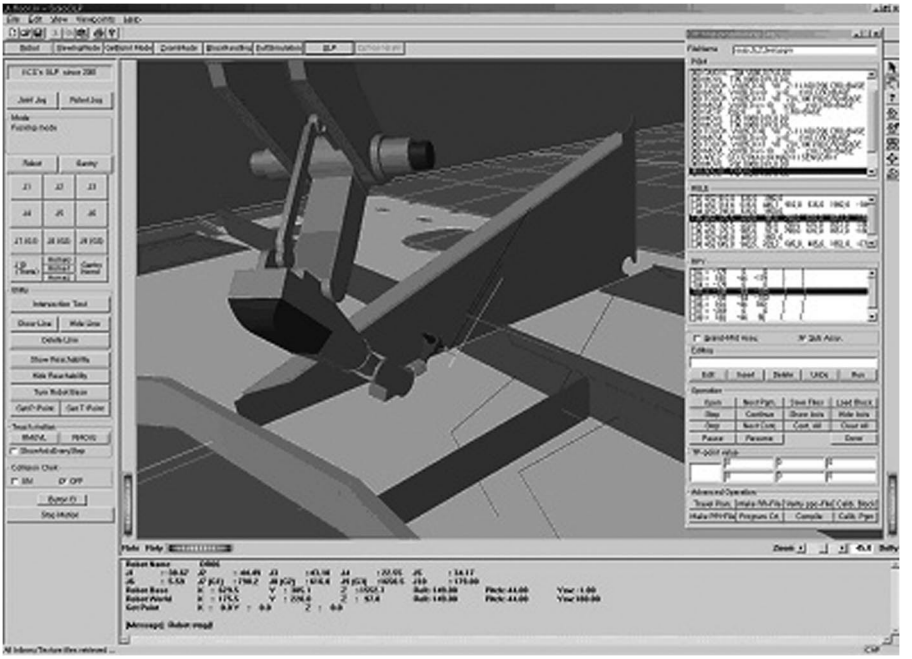


Figure 7. A subassembly simulation.

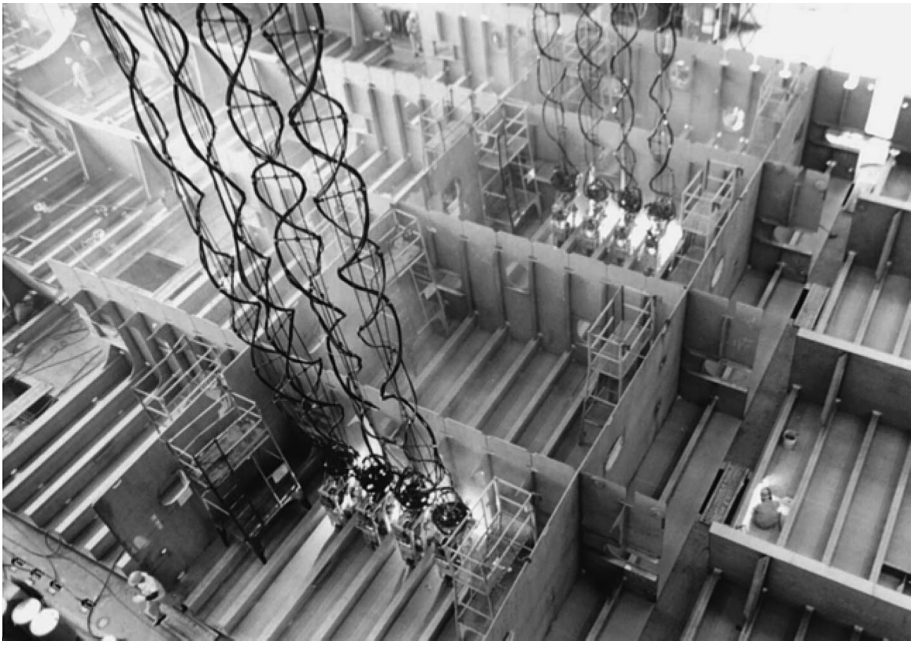


Figure 8. A grand-assembly (real).

multi-threads called for by a 16-ms timer are used for the 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, and that results in memory leakage or malfunction. Thus, the multi-threads and *CCriticalSection* of VC++ work together.

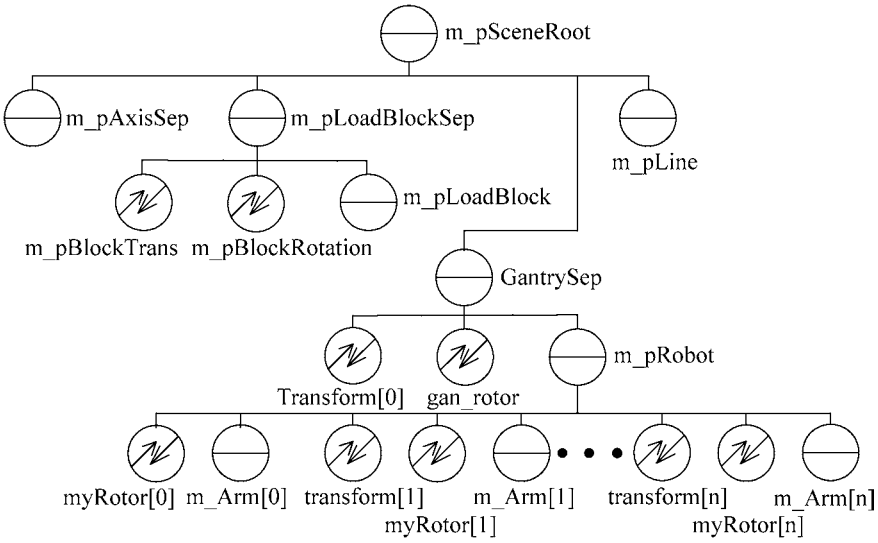
To implement various 3D solid models and motions on a PC, a structured graphical representation of the nodes, illustrated as in Fig. 9a, is required [17–19]. Figure 9b shows the 3D robot body and each link model. Each link can be replaced with a newly designed link without influencing other graphic objects. 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. *m_Arm*[*n*] is a variable name that stores the *n* link model and *myRotor*[*n*] is the variable name of the motion engine. The basic composition 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 ith 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 in Fig. 9. 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* containing the line graphic object is added to the *m_pSceneRoot* node and, whenever the user selects a welding line, the line is displayed in the simulation window.

4. CAD INTERFACE AND PROGRAM GENERATION

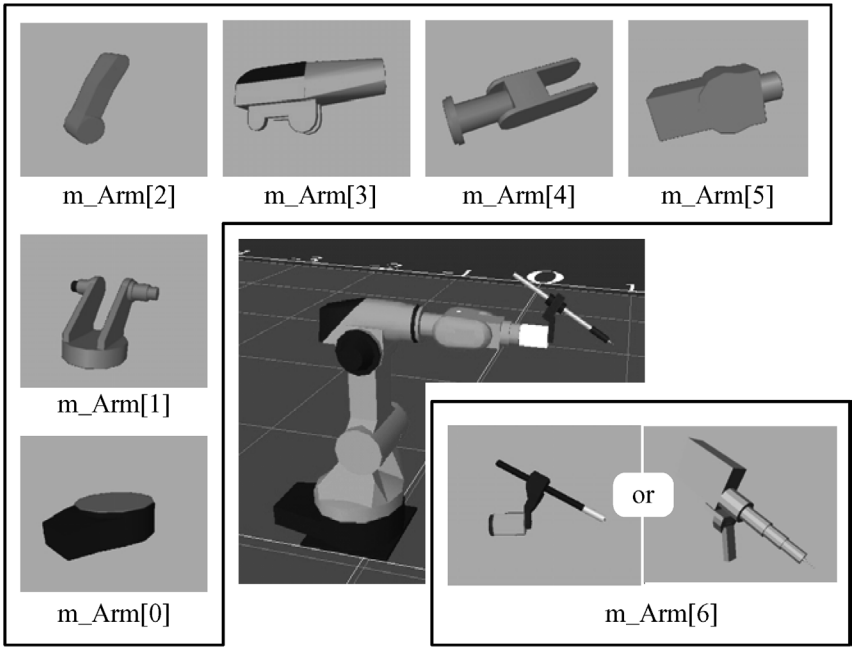
4.1. TRIBON CAD interface

The geometric modeling of robotic systems plays an important role in OLP. A good geometric model of robots, obstacles and the objects manipulated by the robots is important to task planning and path planning. As auxiliary 3D modeling for robot simulations is time consuming and painstaking work, a CAD interface is essential to the robot system.

In developed OLP, 3D geometric models of robot simulations are acquired from a TRIBON CAD interface. The output of the CAD interface is a 3D model of the workpiece that is converted into the VRML. Additionally, the teaching points, which contain the information of the position of the tool center point (TCP) and the orientation of the tool, are also obtained by the CAD interface. Figure 10 shows a simple example of a set of welding information files from the CAD interface.



(a)



(b)

Figure 9. Simulation environment construction: (a) a hierarchical representation of graphic nodes to realize the simulation environment and (b) 3D robot body and individual link models.

A welding information file contains the welding condition that is acquired from experiments and welding standards. It also contains the base coordinate definition, number of welding passes, block names and general block information.

```
BEGIN_INFO
  'S6G9';
  BEGIN_JOINT
    J-1; 527-FR86A-1;527-FR86A-S1;15.5;18.0;
    BEGIN_WELD
    W-1;H;0.707,-0.036,-0.706;525;
      BEGIN_SEGMENT
      1836,-25,314; 0,0,0 1836,500,314;
      END_SEGMENT
    END_WELD
  END_JOINT
END_INFO
```

Figure 10. An example of welding information generated from the CAD interface.

By using the boundary representation (BR) method in Ref. [20], the surface of a solid object is segmented into faces, and each face is modeled by its bounding edges and vertices. Also, the edges and vertices of an object can be extracted from the TRIBON CAD models by the BR method. Therefore, the 3D drawing of TRIBON CAD is decomposed into edges and vertices. The edges and vertices are reconstructed into VRML models. The function *IndexedFaceSet* is a VRML command to render a 3D object from edges and vertices. Using the extracted edges and vertices, all of the elements of the workpiece comprise solid models. Next, these elements are assembled together on the plates according to the assembly's sequence of drawings. This method is similar to the real assembly process in such a way that the operator handles all of the elements in the space and measures the size of each element using simulations, as in a real system.

The CAD interface selects the robot's weldable workpieces and extracts the welding information of the selected workpieces. Weldable workpieces are selected by simulation as well as by human operators. For example, to plan a robot motion trajectory for a line-type seam, the start point and destination point are required. In the case of an arc-type seam, the start point, mid point and destination point are required. Accordingly, the required points are extracted from the vertex datum of the TRIBON CAD according to the {O} coordinate frame in Fig. 4. Because a seam is defined as an adjacent line between two elements, the designated vertexes can be separated from the rest of the vertexes.

The orientation of the torch defined by roll–pitch–yaw values is extracted as shown in Fig. 11, which depicts an example of a specific shape of a workpiece and of the tool orientation values for the welding start and end points. For other shapes of welding seam such as a straight line, a curved line and a vertical line, the normal robot job is accomplished. For the critical examples in Fig. 11, the geometric constraints of a workpiece are: (i) all the elements are made of plate of a thickness of less than 20 mm and (ii) the possible welding length of a robot is longer than 200 mm. After acquiring the welding start and end points of workpieces from the output of the CAD interface where the points are listed line by line in accordance with the sequence of composing the complete object, three adjacent points of all the

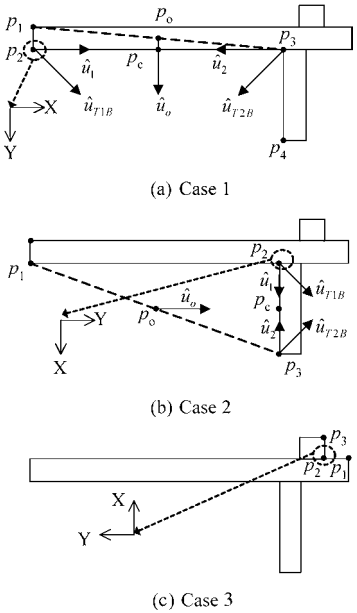


Figure 11. An example of torch pose calculation (cut view of the workpiece). (a) Case 1: convex and available part, (b) Case 2: concave and available part, and (c) Case 3: concave and unavailable part.

edges in a workpiece are gathered as $p_1(x_1, y_1, z_1)$, $p_2(x_2, y_2, z_2)$ and $p_3(x_3, y_3, z_3)$. Then, the center positions p_c and p_o are obtained as:

$$p_c = (p_2 + p_3)/2, \tag{1}$$

$$p_o = (p_1 + p_3)/2. \tag{2}$$

The values p_c and p_o are depicted in Fig. 11. Let the distance between two points $p_A(x_A, y_A, z_A)$ and $p_B(x_B, y_B, z_B)$ be:

$$l(p_A, p_B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2}. \tag{3}$$

The union vectors are obtained as:

$$\hat{u}_o = \frac{\overrightarrow{p_o p_c}}{l(p_o, p_c)}, \tag{4}$$

$$\hat{u}_1 = \frac{\overrightarrow{p_2 p_c}}{l(p_2, p_c)}, \tag{5}$$

$$\hat{u}_2 = \frac{\overrightarrow{p_3 p_c}}{l(p_3, p_c)}. \tag{6}$$

The two vectors are obtained as:

$$\begin{cases} u_{\mathrm{T1B}} = \overrightarrow{u_1} + \overrightarrow{u_o} \\ u_{\mathrm{T2B}} = \overrightarrow{u_2} + \overrightarrow{u_o}, \end{cases} \tag{7}$$

where u_{T1B} and u_{T2B} are the vectors of the tool's direction projected on a plate. Considering the shape of the seam line and the constraints, u_{T1B} and u_{T2B} are translated into real tool direction vectors as in the following four cases: Case 1 is the convex part of a workpiece, Case 2 is the concave part of a workpiece, Case 3 is the impossible shape of the robot's welding and Case 4 is considered as a normal welding seam line.

Case 1: $\{l(p_1, p_2) \leqslant 20$ and $l(p_2, p_3) \geqslant 200\}$ or $\{l(p_1, p_2) \geqslant 200$ and $l(p_2, p_3) \leqslant 20\}$

$$\begin{cases} u_{\mathrm{T1}} = R_{XYZ}\left(\dfrac{\pi}{4}, 0, \pi\right) u_{\mathrm{T1B}} \\ u_{\mathrm{T2}} = R_{XYZ}\left(\dfrac{\pi}{4}, 0, \pi\right) u_{\mathrm{T2B}}. \end{cases} \tag{8}$$

Case 2: $l(p_1, p_2) \geqslant 200$ and $l(p_2, p_3) \geqslant 200$

$$\begin{cases} u_{\mathrm{T1}} = R_{XYZ}\left\{\dfrac{\pi}{4}, 0, sign(\overrightarrow{u_1} \times \overrightarrow{u_o})\dfrac{\pi}{2}\right\} u_{\mathrm{T1B}} \\ u_{\mathrm{T2}} = R_{XYZ}\left\{\dfrac{\pi}{4}, 0, sign(\overrightarrow{u_2} \times \overrightarrow{u_o})\dfrac{\pi}{2}\right\} u_{\mathrm{T2B}}. \end{cases} \tag{9}$$

Case 3: $l(p_1, p_2) < 20$ and $l(p_2, p_3) < 20$,
indeterminate seam.
Case 4: $20 < l(p_1, p_2) < 200$ and $20 < l(p_2, p_3) < 200$
normal welding seam,
where:

$$R_{XYZ}(\gamma, \beta, \alpha) = \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix}.$$

Finally, the tool orientation value of each seam $R_{\mathrm{T1}}(\gamma, \beta, \alpha)$ according to world frame {W} is defined as:

$$\begin{cases} R_{\mathrm{T1}}(\gamma, \beta, \alpha) = {}^{\mathrm{W}}T_{\mathrm{O}} u_{\mathrm{T1}} \\ R_{\mathrm{T2}}(\gamma, \beta, \alpha) = {}^{\mathrm{W}}T_{\mathrm{O}} u_{\mathrm{T2}}. \end{cases} \tag{10}$$

The value of ${}^{\mathrm{W}}T_{\mathrm{O}}$ is shown in Fig. 4.

4.2. Automatic generation of the robot program

In operating welding robots in a shipyard, the most time-consuming aspect is robot programming. In particular, 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 from the CAD interface. First, by the analysis of the shape of the workpiece, the shape can be represented as a simple geometry such as a scallop, a hole, a horizontal line, a horizontal curve, a vertical line, etc. The programs for these simple geometries have already been created and saved in the robot program database. When the workpiece is allocated to the robot system, the robot program generation algorithm divides the workpiece into simple geometries already defined. Next, the robot programs required for the respective simple geometries are selected from the database and combined together to complete the entire program for the given workpiece. The size of the workpiece from the CAD data is reflected in the teaching points in the program.

Figure 12 shows a flowchart representing the automatic robot program generation. The robot program generation algorithm is composed of five subroutines: 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 workplace 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 respective robot programs. In the compilation routine, the matched robot programs are combined into a standard program. The program writing routine rewrites the robot program to fit it to the format of the robot language. Moreover, it sorts the teaching points according to the teaching point's number, and matches each teaching point in the robot program to the respective values in the rule and rpy files.

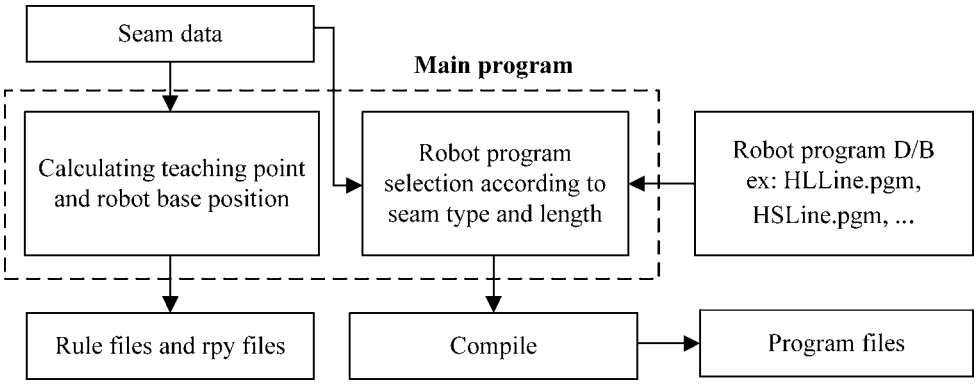


Figure 12. A flowchart for automatic program generation.

5. BLOCK ARRANGEMENT AND PATH GENERATION

5.1. Block arrangement simulation

In order to increase the efficiency of a block arrangement, and reduce the positioning errors between a real arrangement and its simulation, a block arrangement simulation can be performed. Figure 13 illustrates a block arrangement simulation.

The block arrangement simulation helps not only to improve the efficiency of the workplace, but also to correctly position the blocks. In this manner we can arrange as many workpieces as possible in a bounded area. Thereby, block arrangement simulation with 3D block models is very beneficial. The arranged blocks can be printed to help the workers to correctly position the blocks on the workplace. The reference coordinates of the robot program generation are based upon the local coordinate frame {O} of each workpiece depicted in Fig. 4. Therefore, when the robot program is applied to a workplace where many workpieces are scattered, the reference coordinates of each robot program are translated into a world coordinate frame {W}. The translation matrix from {O} to {W} is provided by the results of the block simulation and a vision sensor.

The vision sensor is used to calibrate the difference in the unloading positions of the workpiece between the simulation and the real operation. Figure 14 shows a picture of the vision sensor used. To reduce the vision sensor's marker-searching time, the initial searching position is obtained by the result of the block arrangement

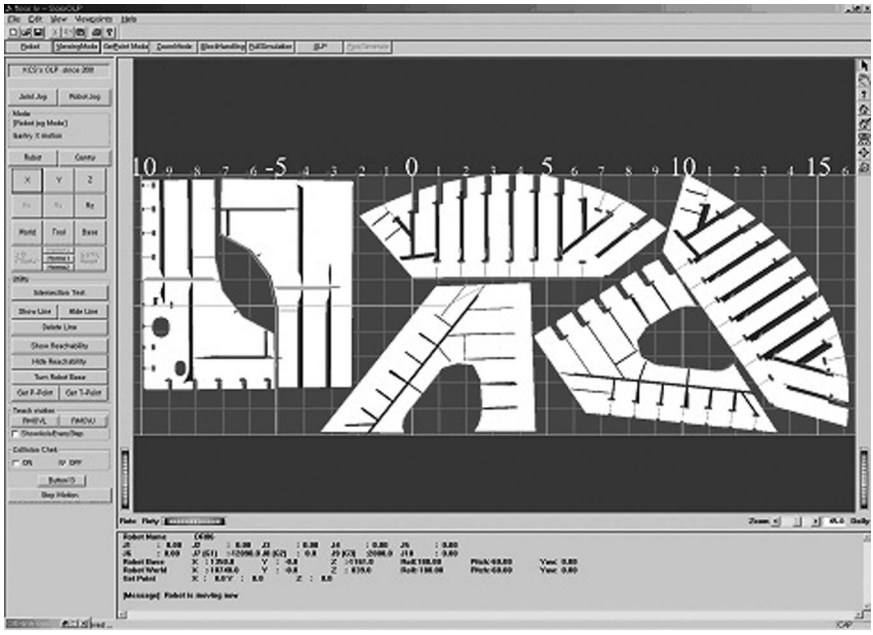


Figure 13. Block arrangement simulation view.

simulation. Once the vision sensor captures the positions of the three markers placed on the workpiece, the block's real position is calculated. Because the vision sensor is placed near the robot base frame {B}, translation only on the xy-plane is possible. The values of translation from the vision sensor to the base frame {B} can be obtained as:

$$\begin{cases} x_{\{B\}} = x_{\text{vision}} + 300 \\ y_{\{B\}} = y_{\text{vision}}, \end{cases} \tag{11}$$

where x_{vision} and y_{vision} are the captured values of each mark according to the vision sensor base, and 300 is the offset of the vision sensor from the base frame {B}. Next, the marker position in reference to the world frame {W} is obtained as:

$$M_{\{W\}} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} = {}^{W}_{B}T \begin{bmatrix} x_{\{B\}} \\ y_{\{B\}} \end{bmatrix}, \tag{12}$$

where M_W is the marker position. The rotation values relative to the {W} frame are calculated using three different M_Ws on the plate. The robot's job program generated for the {O} frame can be transformed into the {W} frame by using the transformation ${}^{W}T_O$ as follows:

$${}^{W}T_O = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & d_x \\ \sin\theta & \cos\theta & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{13}$$

where d_x, d_y and d_z are the position components of the origin of the {O} frame in reference to the {W} frame. In most cases d_z is zero, which means that the block is

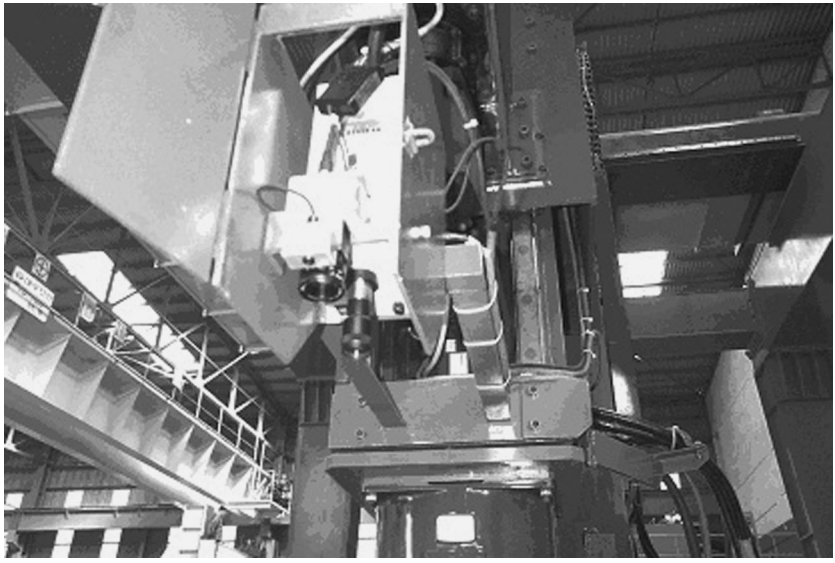


Figure 14. The vision sensor used to calibrate the position error between simulations and real ones.

always placed on the workplace. If d_z is not zero, the operator measures the height of the block's base plate and inputs it to the off-line programs. The origin of the {O} frame and the positions of the markers are predefined by the designer. In the case of a multi-robot system, a calibrated job program is allocated to each robot by a job program distribution algorithm, in which the determining fact is the sum of the lengths of all of the seams in each robot's working area.

5.2. Optimal path generation using a GA

Given that approximately 100–150 seams are randomly distributed on the workplace, the robot's traveling path should be minimal, resulting in the minimum working time. Among many optimization tools, a GA is used for our robot system [21–25]. Travel length minimization is more efficient than working time optimization. There are two reasons for this. First, welding equipment malfunction, a robot's unpredictable motion and interruptions by operators occur frequently while a robot is working; therefore, the time consumed by these problems cannot be considered in the GA calculation and simulation. Second, as the acceleration and deceleration times for trajectory planning are different according to the length of a seam, so the welding time is not exactly known. Alternatively, by assuming that the velocity of the end-effecter is constant, the welding time can be calculated. However, this approach is the same as the travel length minimization by the inverse proportional relationship of time and length. Here, the optimization problem takes the form of a traveling length minimization.

In optimizing a small amount of genetic information or in solving a simple optimization problem, a GA that uses binary bit string encoding is prevalent. However, for welding robots in a shipyard, multi-robot cooperative work with regard to several seam positions, teaching points, variant robots and welding information has to be considered. Accordingly, the use of character-type encoding is suggested. Figure 15 shows the structure of a chromosome. The number of genotypes is the same as the number of seams. The assembly ID, path ID and robot ID are encoded in a genotype.

Frequently, a general crossover method may result in a collision path in the multi-robot cooperative work. The general crossover exchanges a gene of parent 1 with a gene of parent 2. In this case, the exchanged gene of the assigned robot can be different and so the child generated by the parents has to take the collision path. Therefore, we use the partially matched crossover (PMX) suggested in Ref. [22]. The PMX guarantees the validity of the solution of the traveling salesman problem (TSP) presented as a path generation. The children of the PMX inherit some pieces

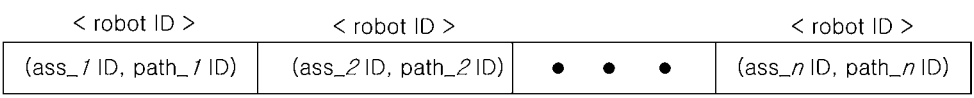


Figure 15. Structure of the chromosome used for robot path planning.

of the genetic information from the first parent by a two-point crossover and the other pieces are inherited according to the gene sequence of the second parent. The possibility of crossover occurrence is determined by the random number of each gene. For example, assume that the parents are defined as P_1 and P_2, the children are defined as C_1 and C_2, and the chromosomes, P_1 and P_2 are initialized as $P_1 = (1\ 2\ 3 \mid 4\ 5\ 6\ 7 \mid 8\ 9)$ and $P_2 = (2\ 4\ 5 \mid 3\ 7\ 6\ 9 \mid 1\ 8)$, where '|' is a cut point; physically, a district separation, for each robot. The number between cut points is an allocated path's number. For the case of P_1, the first, second and third paths are allocated to the first robot, the fourth, fifth, sixth and seventh paths are assigned to the second robot, and the eigth and ninth paths are assigned to the third robot. The '×' is an undefined gene. If the child inherits the second robot's genes, $C_1 = (\times\ \times\ \times \mid 3\ 7\ 6\ 9 \mid \times\times)$ and $C_2 = (\times\ \times\ \times \mid 4\ 5\ 6\ 7 \mid \times\times)$, then the related mapping is $3 \leftrightarrow 4$, $7 \leftrightarrow 5$, $6 \leftrightarrow 6$, $9 \leftrightarrow 7$. By the transitivity rule, the mapping is changed into $3 \leftrightarrow 4$, $9 \leftrightarrow 5$, $6 \leftrightarrow 6$. If genes independent of the mapping are inherited, the children are $C_1 = (1\ 2\times \mid 3\ 7\ 6\ 9 \mid 8\times)$ and $C_2 = (2\ \times\ \times \mid 4\ 5\ 6\ 7 \mid 1\ 8)$. The final children are obtained as $C_1 = (1\ 2\ 4 \mid 3\ 7\ 6\ 9 \mid 8\ 5)$ and $C_2 = (2\ 3\ 9 \mid 4\ 5\ 6\ 7 \mid 1\ 8)$ by the mapping. Concurrently, the assigned genes for each robot are changed with respect to the cut points.

Although the PMX is suitable to prevent a collision path, the crossover method is restricted in the sense that a crossover occurs over a limited range. Hence, the range of solutions is also restricted and a set of solutions does not differ from an initial population, but is constant. To search variable solutions, an adaptive mutation method that enables the exchange of the allocated work by PMX is used. The adaptive mutation method works better than the constant mutation rate method and also prevents collisions among multi-robots. The mutation rate is an important factor determining the convergence and the rate of the convergence of the solution. For a real robot system, because the robot's access to the neighboring seams near the separation layer is easier than its access to distant seams, the mutation rate of seams near the separation layer is high and the mutation rate of seams distant from the separation layer is low. The separation layer is predefined by the block arrangement. Here, we accommodate a Gauss function in (14) as an adaptive mutation rate. Figure 16 shows the Gauss function:

$$y_i = k \exp\left\{-\frac{(x - x_i)^2}{2\sigma^2}\right\}, \tag{14}$$

where $i = 0, \ldots, n$, n is the number of robots, $k = 1$, $\sigma = 0.25$, x_i is the position of the gantry and y_i is the mutation rate. The dominant mutation rate in (14) is determined by k, which is the maximum value, and x, which is obtained when y is more than 68% of the whole area.

We use the rank selection method as a selection routine. First, the chromosomes are sorted by the fitness of each chromosome. Second, the value is calculated as

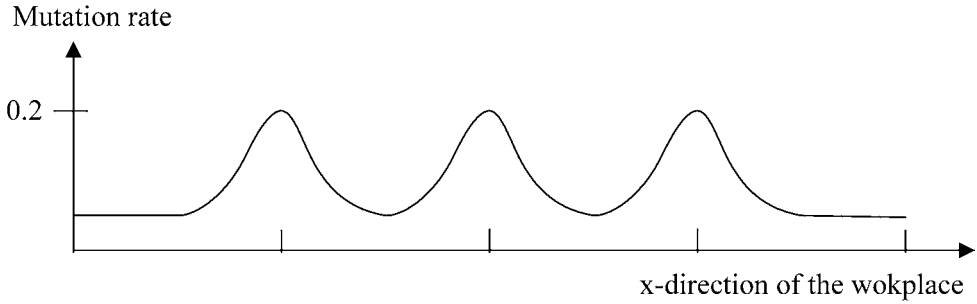


Figure 16. The mutation rate of the spatially adaptive mutation algorithm.

follows:

$$\text{value} = \frac{\text{bias} - \sqrt{\text{bias} \times \text{bias} - 4.0(\text{bias} - 1.0) \times \text{drandom}()}}{2.0 \times (\text{bias} - 1.0)}, \tag{15}$$

where the range is the size of a population and drandom( ) is a function that returns a random value. The position of the selected chromosome is defined as:

$$\text{range} - (\text{base} + \text{value}), \tag{16}$$

where the base is the position of the last chromosome in a population. The selection constraints are assumed as follows: (i) no robot can move beyond its working area restricted by the gantry crane's span, (ii) no collision is permitted and (iii) there is no limit of the seam length for the welding equipment.

In this way, an execution order of the job programs calibrated by the vision sensor is defined. Examples of GA computing for a simple subassembly are shown in Fig. 17. Figure 18 depicts the best fitness value and the average fitness value of each generation for three robots and for a single robot. For the multi-robot system, to allocate workpieces to each robot, we engaged a simple allotment method handled by operators. Two main determinant factors are welding length and collision avoidance. To prevent collision, we restricted the robot's working volume by dividing the workplace into the same numbered zone of the robot. So the workpieces lying on each zone are allotted to each robot system initially. In the GA, initially allotted workpieces are modified partly to justify the welding length of each robot system. As we expected, we can see that three robots are more efficient than a single robot according to the computation time and the best fitness value.

6. IMPLEMENTATION

The tools utilized in this work 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. The Open Inventor was used to implement the graphic environment for the OLP. Because the Open Inventor is a graphic library

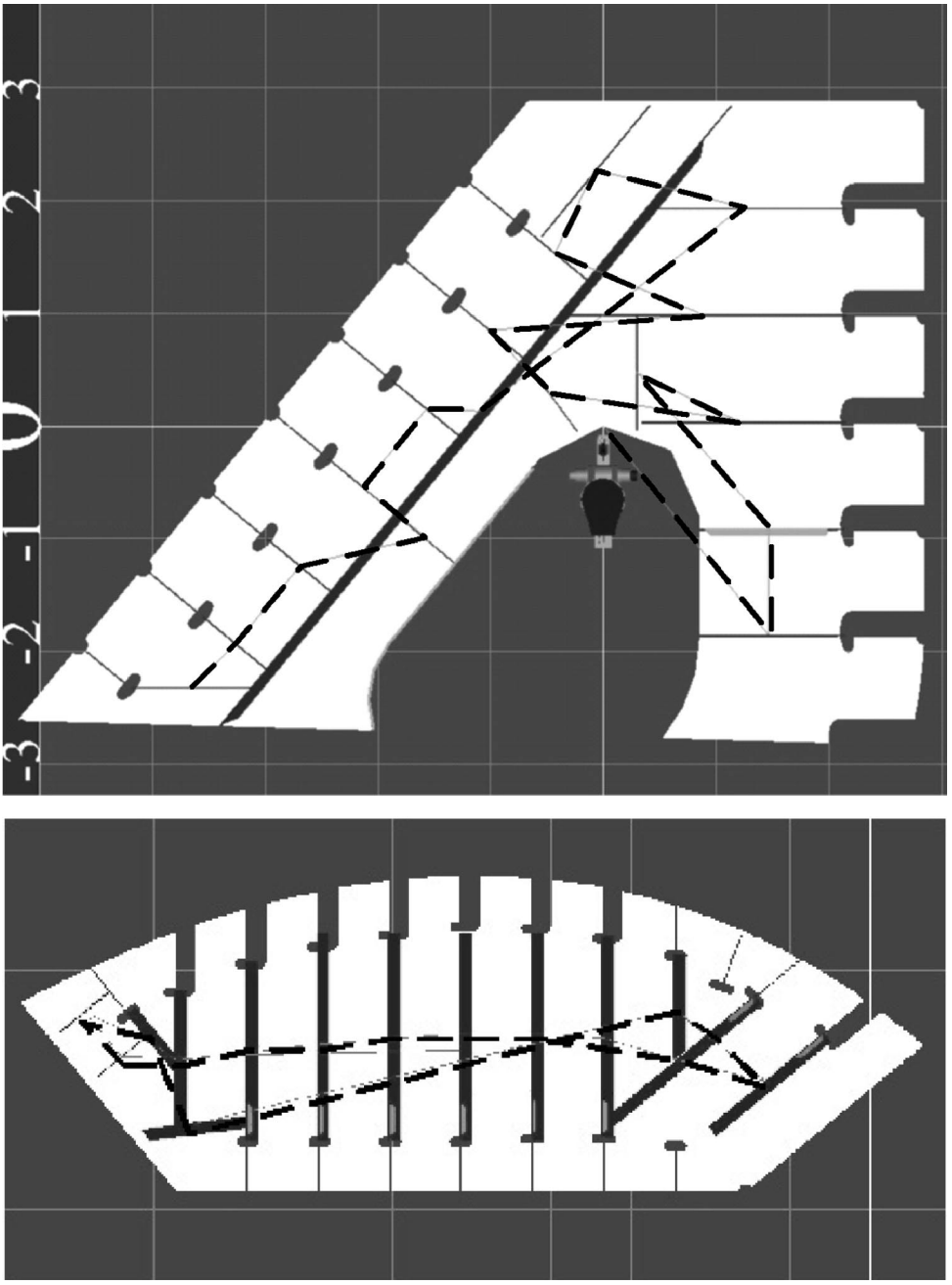
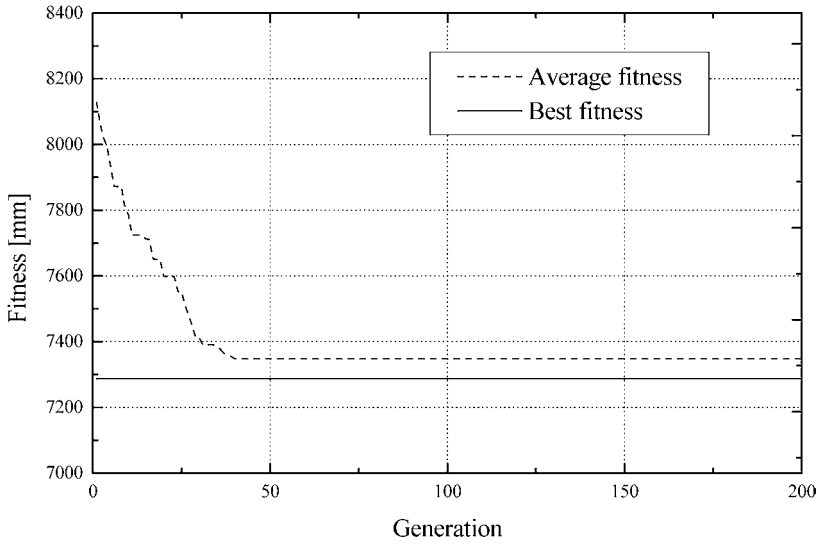
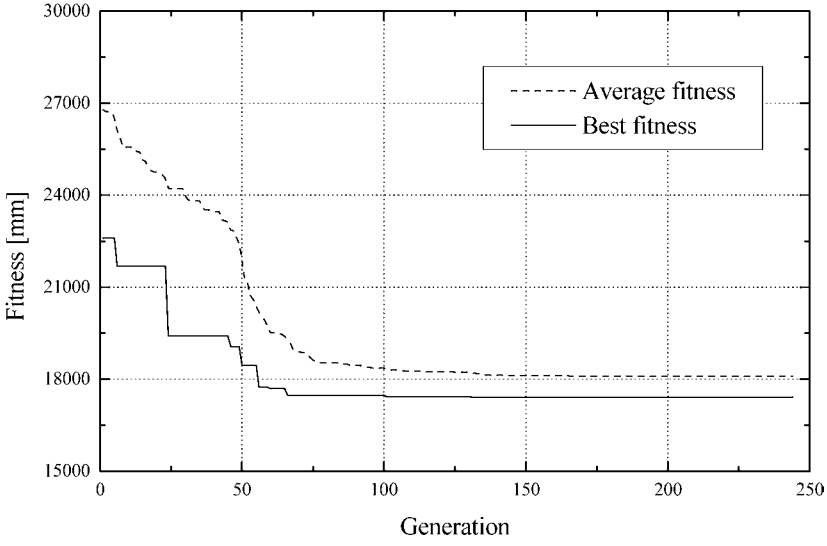


Figure 17. Two examples of an optimal path generated for a single robot system: the dashed line indicates the optimal path represented by the robot base position.



(a) For the case of 3 robots.



(b) For the case of a single robot.

Figure 18. Fitness of the GA algorithm for an example in Fig. 17. The best fitness value of (a) is 7250 mm and the best fitness value of (b) is 17 000 mm. As we expected, the three-robot system is more efficient than the single-robot system. In (a), we can see that due to the best chromosome generated during the initial population computation, the best fitness value is constant over the entire generation.

Table 1.
The efficiency of PC-based OLP compared with on-line teaching

Approximate time to	PC-based OLP	On-line teaching
Generate a standard program	2 s	1 day
Generate all standard programs	5 min	1 week
Generate job programs	5 min	1 h

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. For the subassembly welding robot system, the whole operation of PC-based OLP is used. Table 1 shows the effectiveness of PC-based OLP compared to on-line teaching.

7. CONCLUSIONS

This paper described a PC-based OLP method for welding robots used in the shipbuilding industry. The developed OLP system provides a robot simulation, a block-arrangement simulation, optimal robot traveling path generation and automatic robot program generation. Because graphic environments are made in the VRML, the OLP developed is highly compatible with other software, thus allowing the use of OLP on the Internet. Thereby, adjustments to various robot systems are easy. The OLP developed is very easy for operators to use and maximizes the operating efficiency of welding robot systems in the shipbuilding industry. In the future, due to intelligent robotic techniques such as PC-based OLP, painstaking human labor will be reduced and manufacturing productivity in shipyards will be increased.

Acknowledgements

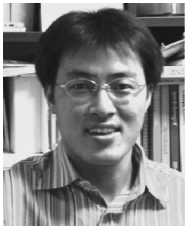
This work was supported by the Ministry of Science and Technology of Korea under a program of the National Research Laboratory, grant NRL M1-0302-00-0039-03-J00-00-023-10.

REFERENCES

1. Y. Aiyama and H. Tanzawa, Management system for distributed and hierarchical robot groups, *Advanced Robotics* **17**, 3–20 (2003).

2. 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* **14**, 333–336 (2000).
3. T. Ogasawara, K. Hashimoto, M. Tabata, M. Komatsu, T. Hara and Y. Kanjo, Application of multi-robots control technology to shipbuilding panels, *J. Robotics Soc., Japan* **16**, 46–47 (1998).
4. J. H. Borm and J. C. Choi, Improvement of local position accuracy of robots for off-line programming, *KSME Int. J.* **6**, 31–38 (1992).
5. R. O. Buchal, D. B. Cherchas, F. Sasani and J. P. Duncan, Simulated off-line programming of welding robots, *Int. J. Robotics Res.* **9**, 31–43 (1989).
6. G. C. Carvalho, M. L. Siqueira and S. C. Absi-Alfaro, Off-line programming of flexible welding manufacturing cells, *J. Mater. Process. Technol.* **78**, 24–28 (1998).
7. M. H. Choi and W. W. Lee, Quantitative evaluation of an intuitive teaching method for industrial robot using a force/moment direction sensor, *Int. J. Control Automat. Syst.* **1**, 395–400 (2003).
8. J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, Reading, MA (1986).
9. J. H. Lee, C. S. Kim and K. S. Hong, Off-line programming in the shipbuilding industry: open architecture and semi-automatic approach, *Int. J. Control Automat. Syst.* **3**, 32–42 (2005).
10. N. Kobayashi, S. Ogawa and N. Koibe, Off-line teaching system of a robot cell for steel pipe processing, *Advanced Robotics* **12**, 327–332 (2001).
11. L. F. Nielsen, S. Trostmann, E. Trostmann and F. Conrad, Robot off-line programming and simulation as a true CIME-subsystem, in: *Proc. IEEE Int. Conf. on Robotics and Automation*, Nice, pp. 1089–1094 (1992).
12. G. Ferretti, S. Filippi, C. Maffezzoni, G. Magnani and P. Rocco, Modular dynamic virtual-reality modeling of robotic systems, *IEEE Robotics Automat. Mag.* **6**, 13–23 (1999).
13. K. S. Hong, K. H. Choi, J. G. Kim and S. Lee, A PC-based open robot control system: PC-ORD, *Robotics Comp. Integrated Manufact.* **17**, 355–365 (2001).
14. B. Kreuzer and D. Milojevic, Simulation tools improve planning and reliability of paint finishing lines, *Industrial Robot* **25**, 117–123 (1998).
15. T. L. Kunii and Y. Shinagawa, *Modern Geometric Computing for Visualization*. Springer-Verlag, Berlin (1992).
16. G. Richard, Object-oriented programming for robotic manipulator simulation, *IEEE Robotics Automat. Mag.* **4**, 21–29 (1997).
17. C. M. Hoffmann, *Geometric and Solid Modeling: An Introduction*. Morgan Kaufmann, San Mateo, CA (1989).
18. TECNOMATIX Technologies, *ROBCAD User Manual*. TECNOMATIX, Herzeliya (1998).
19. J. Wernecke, *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor*. Addison Wesley, Menlo Park, CA (1994).
20. P. C.-Y. Sheu and Q. Xue, *Intelligent Robotic Planning Systems*. World Scientific, Singapore (1993).
21. Y. Davidor, *Genetic Algorithms and Robotics: A Heuristics for Optimization*. World Scientific, Singapore (1991).
22. C. M. Fonseca and P. J. Flemming, Genetic algorithms for multi objective optimization, *Evol. Comput.* **3**, 1–16 (1995).
23. D. Goldberg and R. Lingle, Alleles, loci, and the TSP, in: *Proc. 1st Int. Conf. on Genetic Algorithms*, Hillsdale, NJ, pp. 154–159 (1985).
24. S. R. Munasinghe, M. Nakamura, S. Goto and N. Kyura, Trajectory planning for industrial robot manipulators considering assigned velocity and allowance under joint acceleration limit, *Int. J. Control Automat. Syst.* **1**, 68–45 (2003).
25. I. Namgung, Application of quadratic algebraic curve for 2D collision-free path planning and path space construction, *Int. J. Control Automat. Syst.* **2**, 107–117 (2004).

ABOUT THE AUTHORS



Chang-Sei Kim received the BS degree in Control and Mechanical Engineering from Pusan National University in 1998 and the MS degree in Mechanical Design and Production Engineering from Seoul National University in 2000. Since 2000, he has been with the Robot Research and Development Team in Daewoo Shipbuilding and Marine Engineering (DSME), Korea, as a Researcher. In DSME, he has worked on the development of welding robots, painting robots and factory automation. His current research interests include non-linear robust control, robotic systems, hydraulic systems and control system applications.

Keum-Shik Hong received the BS degree in Mechanical Design and Production Engineering from Seoul National University in 1979, the MS degree in Mechanical Engineering from Columbia University, New York in 1987, and both the MS degree in Applied Mathematics and the PhD degree in Mechanical Engineering from the University of Illinois at Urbana–Champaign (UIUC) in 1991. From 1991 to 1992, he was a Postdoctoral Fellow at UIUC. He joined the School of Mechanical Engineering at Pusan National University, Korea in 1993, where he is now a Professor. During 1982–1985, he was with Daewoo Heavy Industries, Incheon, Korea, where he worked on vibration, noise and emission problems of vehicles and engines. He served as an Associate Editor for the *Journal of Control, Automation and Systems Engineering*, and has been serving as an Associate Editor for *Automatica* (2000–date) and as an Editor for the *International Journal of Control, Automation and Systems* (2003–). He also serves as an Associate Editor on various IEEE and IFAC Conference Editorial Boards. His laboratory, the Integrated Dynamics and Control Engineering Laboratory, was designated as a National Research Laboratory by the Ministry of Science and Technology of Korea in 2003. He received the Fumio Harashima Mechatronics Award in 2003. He is a member of the ASME, IEEE, ICASE, KSME, KSPE, KIEE and KINPR. His current research interests include nonlinear systems theory, adaptive control, distributed parameter system control, input shaping, vehicle control and innovative control applications to engineering problems.

Yong-Seop Han received the BS degree in Metallurgical Engineering from Seoul National University in 1978, the MS degree from KAIST, Korea, and the DrIng degree in Mechanical and Electronic Engineering from T. U. Braunschweig in Germany. Since 1983, he has been with Daewoo Shipbuilding and Marine Engineering, Korea as a Head of the Industrial Application R&D Institute, where he has worked on the development of welding robot systems in assembly stages in the shipyard. His current research interests include welding processes, welding automation and robotic applications in shipbuilding.