

# Welding Robot Applications in Shipbuilding Industry: Off-Line Programming, Virtual Reality Simulation, and Open Architecture

Chang-Sei Kim<sup>2</sup>, Keum-Shik Hong<sup>2</sup>, Yong-Sub Han<sup>1</sup>  
<sup>1</sup>*Daewoo Shipbuilding and Marine Engineering Ltd, South Korea*  
<sup>2</sup>*Pusan National University, South Korea*

## 1. Introduction

The shipbuilding industry is steadily advancing by introducing robots to its work fields for increases in productivity and improvements in working conditions (Nagao et al, 2000). However, the shipbuilding company still faces with the worker's health problem, an increase of aging workers, a shortage of skilled workers, and environmental protection related issues. Therefore, advanced robotic manipulator is still required to overcome these problems. And, how to apply commercial robotic system properly to meet the production purpose in the shipyard is a key research topic for shipbuilding engineers.

The shipbuilding process is mainly divided into design, cutting, welding, assembling, grinding, blinding, and painting process. Among these manufacturing processes, welding is the most crucial, expensive, and time-consuming process. For that reason, welding robot applications have yielded a big productivity improvement in hull assembly welding and have reduced work-related musculoskeletal disorders of workers.

In this chapter, the results of our work in the development of a welding robot system and a PC-based off-line programming for welding robot application on assembly lines in shipbuilding are explained. Also, a methodology of implementing PC-based off-line programming on users PC is presented. The off-line programming is a system that comprises robot's simulation, robot programming, and other functions such as monitoring, scheduling, etc, that makes users operate robot system easily. The off-line programming is essential for welding robot system in shipyard to prepare robot program and then to shorten production time.

Currently, the operation of industrial robots is through either on-line teaching or off-line programming (Choi & Lee, 2003; Carvalho et al., 1998; Craig, 1986;). On-line teaching is, by definition, a technique of generating robot programs using a real robot system, whereas off-line programming 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. But, in such work places as shipbuilding, where the shape and size of workpiece are various (i.e., there are more than 1200 different shapes of workpieces for grand-assembly, if we account the size of these different shaped workpiece, we may not count the different kind of workpieces. Moreover, the new shape of workpiece is still increasing according to the ship specification

Source: Industrial Robotics: Programming, Simulation and Applicationl, ISBN 3-86611-286-6, pp. 702, ARS/pIV, Germany, December 2006, Edited by: Low Kin Huat

advancing), on-line teaching will cause problems due not only to the decrease of productivity caused by halting the robots while reprogramming, but, more importantly, to 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 off-line programming, while only applying programs that were already verified to be effective for the job. The advantages of using off-line programming are: (1) the effective programming of robot-command logic with debugging facilities; (2) the easy verification of the validity of the robot programs through simulation and visualization; (3) organized documentation through simulation models with appropriate programs; (4) the reuse of existing robot programs and easy application to other objects; and (5) the cost independence of production owing to the fact that production can be continued while programming. Fig. 1 shows difference between two robot programming.

Research on the robotic system simulation is prevalent (Ferretti, 1999; Kreuzer & Miojevic, 1998; Richard, 1997). And the industrial robot production enterprises provide commercialized software for robot simulations such as ROBCAD, IGRIP, and etc., 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, the complete modeling of the work object, and the development of language translators that will work with robot manufacturing

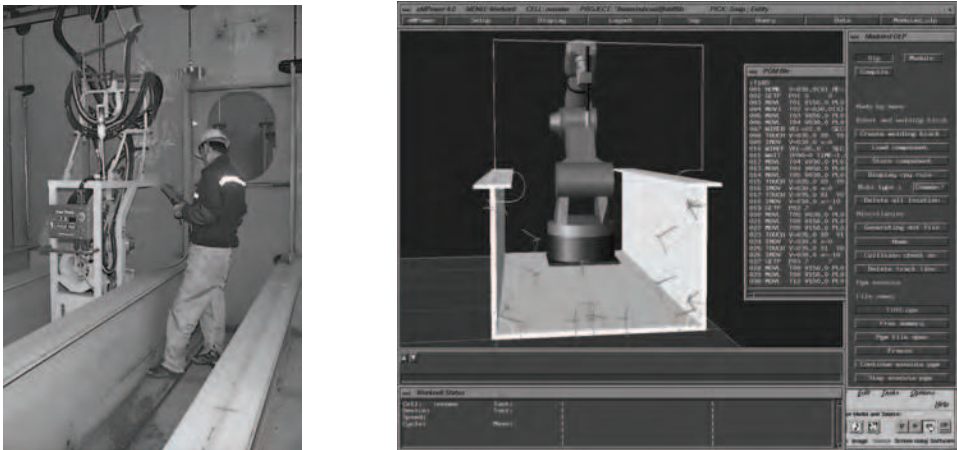


Fig. 1. On-line teaching (Left) and Off-line programming (Right)

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 off-line programming 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.

This chapter is structured as follows: In Section 2, a welding workpiece and a robot system in hull assembly line is explained. In Section 3, the off-line programming and the methodology of robot simulation based on PC is discussed. Section 4 presents the

application of the CAD interface and the robot program generation to the actual welding plant. In Section 5, the process of path planning for scattering the blocks based on a block arrangement simulation is explained. Section 6 provides conclusions.

## 2. Welding Robot System Configuration

Hull assembly is separated to sub-assembly, middle-assembly and grand-assembly by its size and production sequence. One of the critical problems for applying robots to shipbuilding is transferring method to locate robots in a proper position. There are some researches about the shipbuilding process, where the positioning system is a key technology for effective introduction of robots because the workpiece is diverse and complex (Ogasawara et al., 1998; Nagao et al., 2000; Mori et al., 2001).

Currently, there are three methods of transferring a welding robot to a work place: a manual transfer, a semi-automatic transfer and a fully automatic transferring approach. The manual transfer means that the robot is placed on a workpiece by human. The semi-automatic approach, that is good for high assembly blocks, is composed of x-y plane gantry crane, chain-rope, robot cart, and a robot body. Sometimes, robot origin transfer unit is attached to move robot body to its origin position in a workpiece. The semi-automatic type of robot transferring system hangs the robot body at the end of chain-rope. When working, the robot body is lifted up and down by the chain-rope and moves plenary on a workplace by x-y plane gantry crane. Fig. 2 shows a picture of the real grand-assembly welding robot system and a typical example of a workpiece. A 3-axis Cartesian robotic manipulator operates the fully automatic transfer system. The fully automatic transfer system is applied to the sub-assembly lines.

The manual and fully automatic approaches are not preferred in a grand- and middle-assembly process. The reasons are that a manual move consumes too much time and excessive human effort for individual transferences, and that a fully automatic transference causes an unwanted vibration of the gantry, in moving the welding robot from one place to another, which subsequently deteriorates the welding quality. Accordingly, a semi-automatic approach is widely used. However, due to the small size of sub-assembly, the fully automatic transfer approach is applied to the sub-assembly welding process. Recently, new types of robot transferring systems such as mobile type, rail type, and parallel mechanism, are researched for shipbuilding.

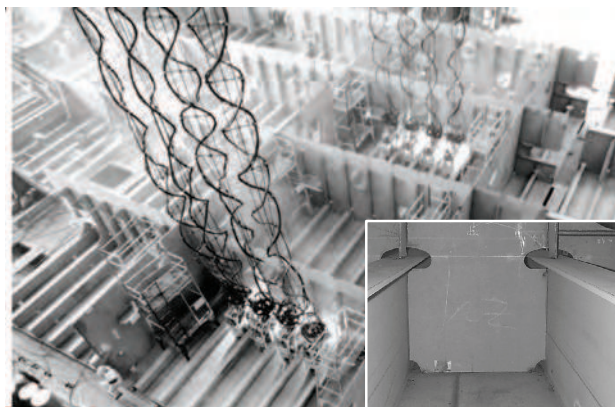


Fig. 2. A grand-assembly and a typical example of a workpiece(Right corner).

Fig. 3 shows the welding robot system that is being used at Daewoo Shipbuilding and Marine Engineering Ltd., South Korea. The system is composed of a welding robot, a controller, a gantry crane, welding equipment, and an on-site industrial computer.

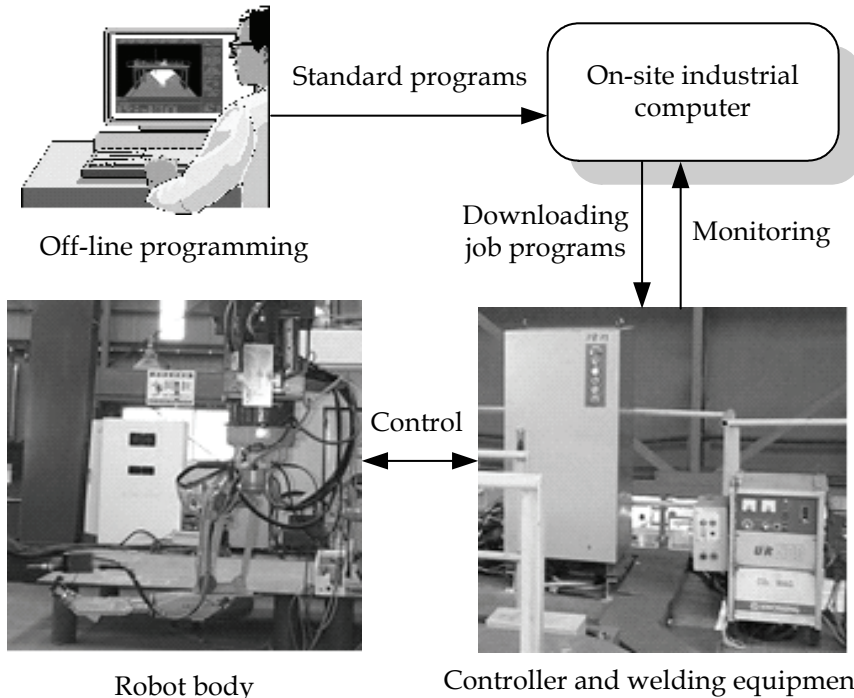


Fig. 3. Configuration of the welding robot system: First, The standard-program is made using the off-line programming. Second, using the CAD interface, the job program which contains real size of a workpiece is made. Third, the job-program is transferred to robot controller. Finally, the robot controller execute the job-program line by line and controls welding robots.

Because the size and shape of workpieces in the shipbuilding industry vary greatly, a 6-axis articulated robot is generally used for welding. The robot controller consists of a Pentium II processor, three motor interface boards and other digital signal processing boards. As one motor interface board can control 4 motors, room for adding more control boards for auxiliary actuators has been reserved. As the controller are designed by open-architecture, regarding the workpiece, auxiliary robotic manipulator to locate robot system or to transfer workpieces can cooperated with the robot's controller. The QNX is used as a real-time operating system.

To increase robot's welding ability against various shape of workpice, we define 2 robot programs as a standard-program and a job-program. And, we design the standard-program as a combination 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 program file. And the rpy-file contains the orientation values of the teaching points.

```

;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
    
```

Fig. 4. An example of the standard program consisting of three parts: a program-file (top), a rpy-file (middle), and a rule-file (bottom). In the program-file, the second column is robot command and the third column is inner paramters of each robot command.

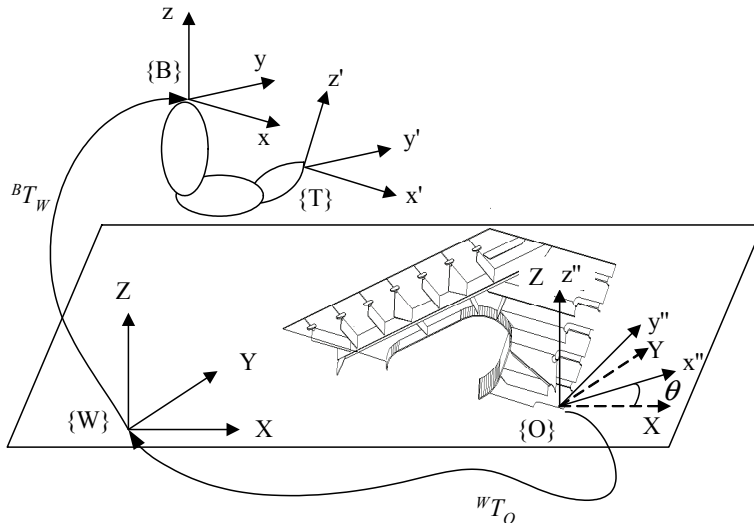


Fig. 5. The coordinate systems defined: world, base, object, and tool.

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. Fig. 4 shows an example of a standard-program. Also, in order to be executed in the robot controller, 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 a job-program. The size information is obtained by CAD interface and the welding conditions are gathered from a database in which all previous experimental data have been stored.

Fig. 5 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. Off-Line Programming and Robot Simulation

#### 3.1 Off-line programming for shipbuilding

The developed off-line programming system is composed of CAD interface, robot simulation, automatic robot program generation, block arrangement simulation, and a path planning using Genetic algorithm for fully automated welding robot system of a shipbuilding. Fig. 6 shows a flow chart of the developed off-line programming system. First, if the workpiece is identical to an old one, then an appropriate program is automatically loaded from the D/B. If the workpiece is not the same as old one, the off-line program generates robot's new programs automatically. Second, if the robot programs are new one, then we check it by simulation. Simulations using off-line programming 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

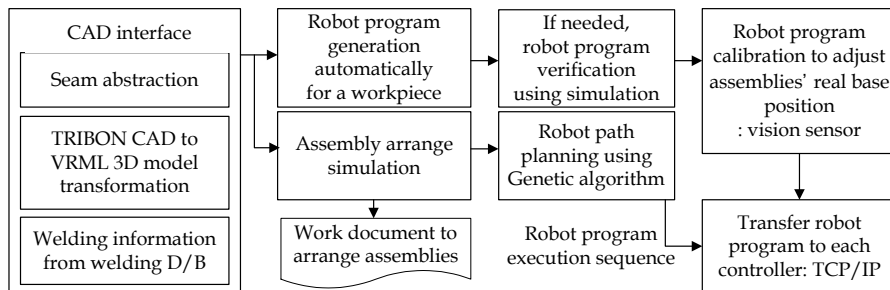


Fig. 6. Flow chart of the off-line programming.

pre-determined surfaces. Third, depending on the work schedule, off-line programming provides a work-order document of a workplace arrangement for the blocks. Fourth, 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 generated robot programs to the each robot's controller using TCP/IP communication. For the grand- and middle-assembly line, because the robot body is located on a workpiece by semi-automatic method, and the robot base position is not changing until robot finishes a welding of a workpiece, the

vision sensor and block arranging to get information of workpiece base position is not required.

### 3.2 Robot body and workpiece modelling using VRML

Using the virtual reality modeling language (VRML), various simulations such as robot motion, block arrangement, optimal trajectory planning, and CAD interfaces were performed on a Personal Computer (PC). The VRML utilized 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 (Ferretti et al., 1999; Kunii & Shinagawa, 1992). Fig. 7 shows examples of VRML models.

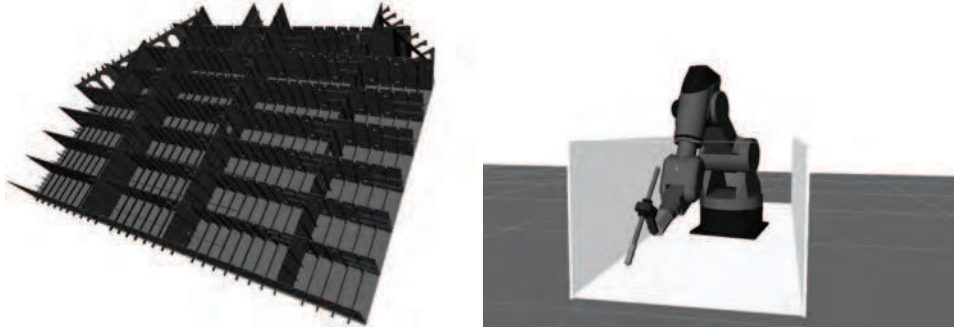


Fig. 7. Examples of VRML 3D models(Left: a grand-assembly model converted from CAD data using developed converting software; Right: a welding robot model drawn by manually using VRML).

The salient features of the VRML are as follows: first, it is easier to interpret because the text is expressed with 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, off-line programming through the Internet is an option.

### 3.3 Kinematics and robot motion structure

To implement robot's 3D solid models and motions on a PC, a structured graphical representation of the nodes, illustrated as in Fig. 8(a), is required (Hoffmann, 1989; TECHNOMATIX, 1998; Wernecke, 1994). Also, Fig. 8(b) 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. The  $m\_Arm[n]$  is a variable name that stores the  $n$ -th link model, and the  $myRotor[n]$  is the variable name of the motion engine. A translation variable  $transform[n]$  is used to place each link on a desired position, i.e. end of the  $n$ -1th link. The basic composition is a parallel combination of the translation variable, motion engine and the link model of each link. Accordingly, the motion of the  $(i + n)$ th link, where  $n = 1, 2, 3, \dots$ , is affected by the movement of the  $i$ -th link. For 6-axis welding robot, as the robot body is composed of 6 rotation links and the value of

$myRotor[n]$  means each link's angle, the robot simulation can be done by changing  $myRotor[n]$  value using robot's forward kinematics and inverse kinematics. The  $m\_pLoadBlockSep$  is added to this node structure to draw workpieces. Usually, There are 2 kinds of simulation option is required for shipbuilding as below:

Case 1: Grand- and Middle-assembly

- Because the workpiece base position is fixed,
- Robot base position is also fixed.

Case 2: Sub-assembly

- Because the workpiece base position is not fixed (variable),
- Robot base position is also movable (variables).

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 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 the  $m\_pLoadBlockSep$  and the  $GantrySep$ , as shown in Fig. 8. The axis graphic node is added to the  $m\_pSceneRoot$  node, whenever the user generates new teaching points. 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. In order to render and change link parameter easily for a similar type robot, a robot initialization file is used. The robot initialization file contains the file name of the robot body VRML model, the link parameters, the limit values of individual joints, the control parameter 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-effector and possible collisions with the surrounding parts can be easily examined through the simulations in a virtual environment.

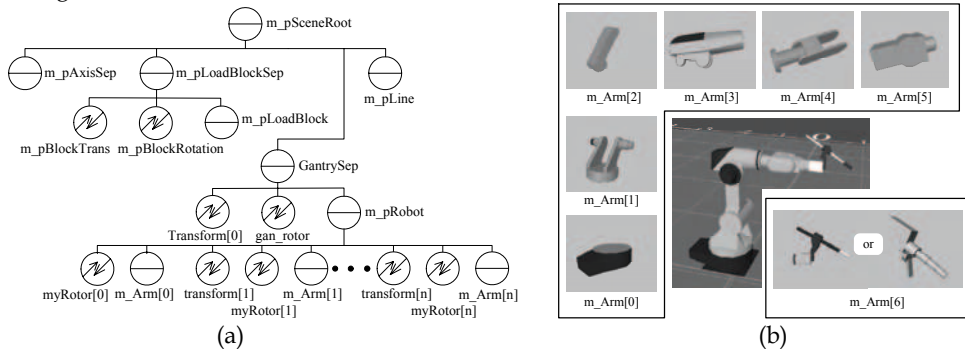


Fig. 8. Simulation environment construction: (a) a hierarchical representation of graphic nodes to realize simulation environment; (b) 3D robot body and individual link models.

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 to change teaching point on simulation window. 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. 5. In the program execution mode, robot commands in a standard program are automatically, simultaneously, and continuously executed line by line. Left picture of Fig. 9 shows the simulation of a grand-assembly, while right picture shows the simulation of a sub-assembly for which the robot is the hanging-type.

The simulated motions approach to 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 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 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. So the multi-threads and *CCriticalSection* of VC++ work together.

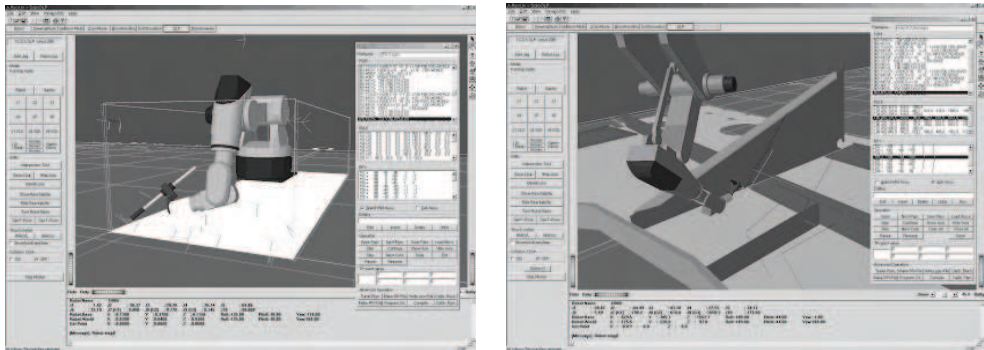


Fig. 9. Simulation example of a welding robot system(Left: for a grand- and middle-assembly, Right: sub-assembly).

## 4. CAD Interface and Automatic Robot Program Generation

### 4.1 Tribon CAD interface

The geometric modeling of robotic systems plays an important role in off-line programming. 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 off-line programming, 3D geometric models of robot simulations are acquired from a TRIBON CAD interface. TRIBON CAD is commercial software used as a design tool of shipbuilding. The output of the CAD interface is a 3D model of the workpiece that is converted into the VRML. Additionally, the welding information, which contains the position value of the tool center point (TCP) and the orientation of the tool on the workpiece, are also obtained by the CAD interface. Fig. 10 shows a simple example of a set

of welding information file from CAD interface. A welding information file also contains the welding condition that is specified by class and welding standards. Also it contains general block information such as the number of welding passes, base coordinate definition, block names, and block sizes.

By using the boundary representation (BR) method (Sheu & Xue, 1993), 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.

```

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

```

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

Using the CAD interface, the operator can select the robot weld-able workpieces and extracts the welding information of the selected workpieces on operator's PC. For the case of grand- and middle-assembly, as the robot weld-able workpiece type is pre-defined by operator and designer, only the workpiece size is required to make job-programs. However, for the sub-assembly, robot weld-able workpiece is not pre-defined, the operator have to select robot's workpiece and input not only size but also shape of the workpiece. So, CAD interface works more important role for the sub-assembly line. For example, to plan a robot motion trajectory of a line-type seam in a sub-assembly, the start point and destination point are required. In the case of an arc-shape 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 in according to the {O} coordinate frame in Fig. 5. 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

straight line, curved line, and vertical line, normal robot job is accomplished. For the critical examples in Fig. 11, the geometric constraints of a workpiece are: (1) all the elements are made of plate of a thickness of less than 20 mm; (2) 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 CAD interface where the points are listed line by line in accordance to the sequence of composing complete object, 3 adjacent points of all the edge in a workpiece are gathered as  $p_1(x_1, y_1, z_1)$ ,  $p_2(x_2, y_2, z_2)$ , and  $p_3(x_3, y_3, z_3)$ . And then, the center positions  $p_c$  and  $p_o$  are obtained as (1) and (2).

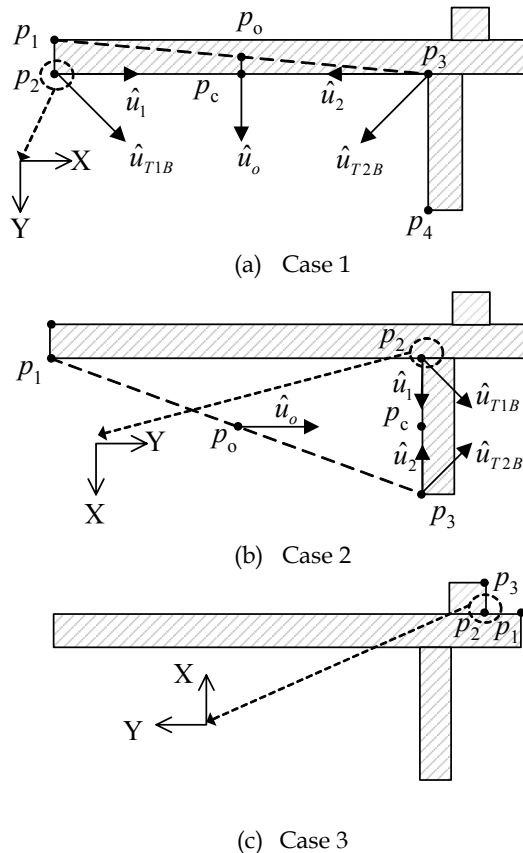


Fig. 11. An example of torch pose calculation (cut view of the workpiece). (a) Case 1: convex and available part; (b) Case 2: concave and available part; (c) Case 3: concave and unavailable part.

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

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

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

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

The union vectors are obtained as

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

$$\hat{u}_1 = \frac{\overrightarrow{p_1 p_c}}{l(p_2, p_c)} \quad (5)$$

$$\hat{u}_2 = \frac{\overrightarrow{p_2 p_c}}{l(p_3, p_c)} \quad (6)$$

The two vectors are obtained as

$$\begin{cases} u_{T1B} = \overrightarrow{u_1} + \overrightarrow{u_o}, \\ u_{T2B} = \overrightarrow{u_2} + \overrightarrow{u_o}, \end{cases} \quad (7)$$

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 4 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:**  $\{l(p_1, p_2) \leq 20 \text{ and } l(p_2, p_3) \geq 200\}$  or  $\{l(p_1, p_2) \geq 200 \text{ and } l(p_2, p_3) \leq 20\}$

$$\begin{cases} u_{T1} = R_{XYZ}(\frac{\pi}{4}, 0, \pi) u_{T1B}, \\ u_{T2} = R_{XYZ}(\frac{\pi}{4}, 0, \pi) u_{T2B}. \end{cases} \quad (8)$$

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

$$\begin{cases} u_{T1} = R_{XYZ}(\frac{\pi}{4}, 0, \text{sign}(\overrightarrow{u_1} \times \overrightarrow{u_o}) \frac{\pi}{2}) u_{T1B}, \\ u_{T2} = R_{XYZ}(\frac{\pi}{4}, 0, \text{sign}(\overrightarrow{u_2} \times \overrightarrow{u_o}) \frac{\pi}{2}) u_{T2B}. \end{cases} \quad (9)$$

**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 seam.

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

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

$$\begin{cases} R_{T1}(\gamma, \beta, \alpha) = {}^W T_O u_{T1}, \\ R_{T2}(\gamma, \beta, \alpha) = {}^W T_O u_{T2}. \end{cases} \quad (10)$$

The value of  ${}^W T_O$  is shown in Fig. 5.

In this way, the teaching points of newly introduced seams can be calculated using the information of CAD interface. Also, because the once defined rules can be adapted to other similar cases, newly rule-adding works are decreasing.

#### 4.2 Automatic generation of robot program

In operating welding robots in a shipyard, the largest of time consumption is in robot programming. Particularly, for the 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 gathered 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 and a vertical line, and others. The programs for these simple geometries are already 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 simple geometries already defined. 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 the given workpiece. Fig. 12 shows an example of workpiece composition using simple geometry method. The size of the workpiece from the CAD data is reflected in the teaching points in the program.

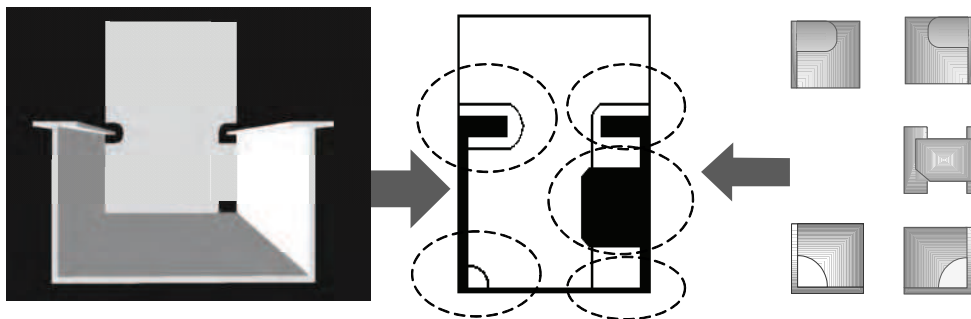


Fig. 12. An example of workpiece composition for a grand-assembly.

Fig. 13 is a flow chart representing the 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 and no singular occur in moving from one teaching point to another teaching point. The nonsingular and 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 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 the teaching point's number and matches each teaching point in the robot program to the respective values in the rule- and rpy- files.

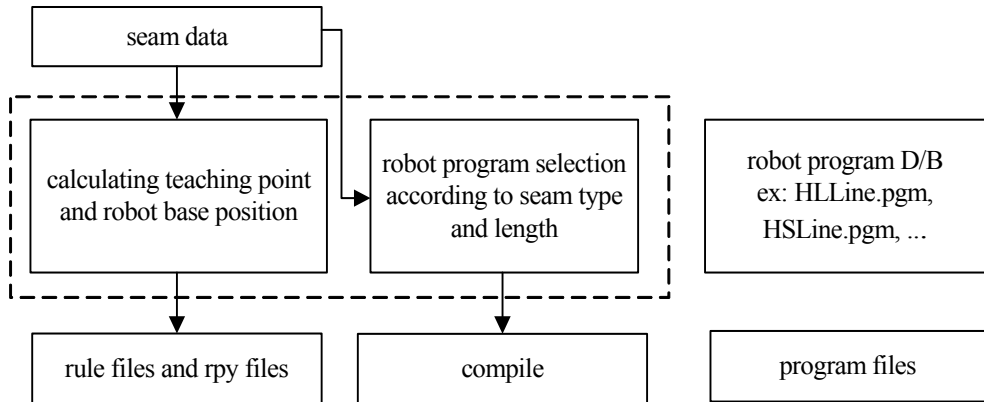


Fig. 13. A flow chart for automatic program generation.

## 5. Block Arrangement and Path Generation

For the grand- and middle-assembly, the assembly block position is fixed on a workplace, the block arrangement and path generation is not required. But, for the sub-assembly, as the block can be placed on a workplace by operator using over-head crane, we can plan the block arrangement by the simulation to predict the base position of each workpiece. Also, for the fully automated robotic system of a sub-assembly line, robot's path planning to minimize travel length is very important to reduce working hours.

### 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, block arrangement simulations are performed on the off-line programming. Fig. 13 illustrates a block arrangement simulation view for the sub-assembly line.

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, the block arrangement simulation with 3D block models is very beneficial to estimate the amount of work per a workplace and job scheduling. Also, the arranged block position and drawing can be printed out to a document that helps the workers to place the blocks on a workplace. The reference coordinates of the robot program generation are based upon the local

coordinate frame {O} of each workpiece depicted in Fig. 5. Therefore when the robot program is applied to a workplace where many workpieces are scattered such as Fig. 14, 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.

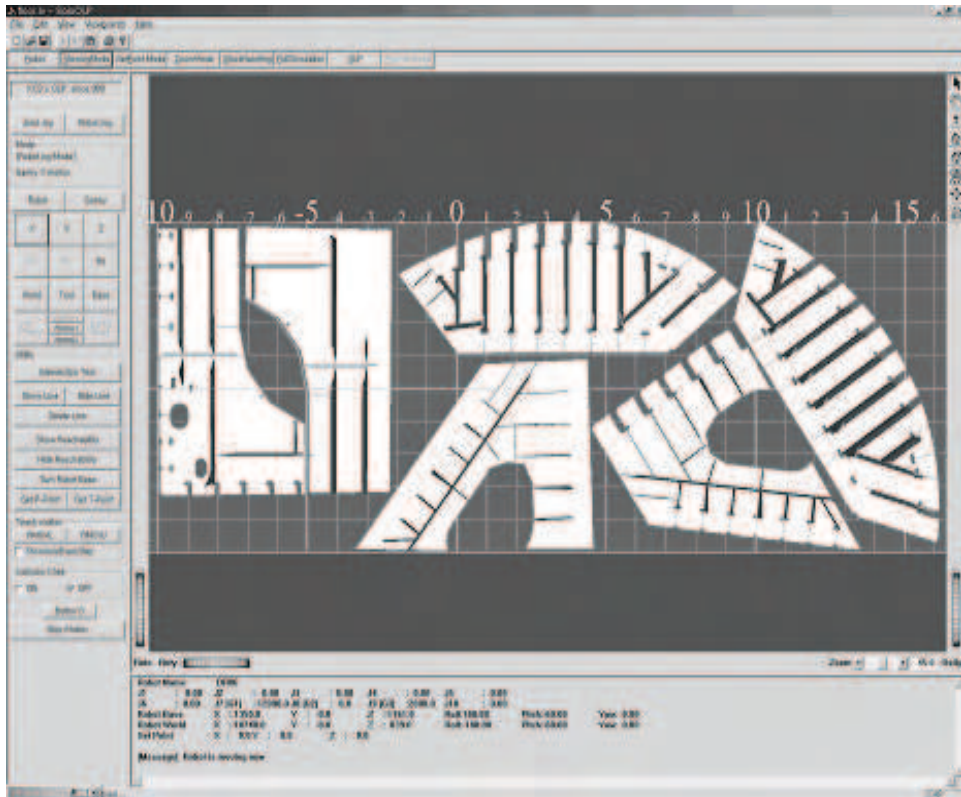


Fig. 14. Block arrangement simulation view.

The vision sensor is used to calibrate the unloading position of workpieces difference between the simulation and the real operation. The operation of the vision sensor is searching marker on a workpiece and marker's position is pre-defined. To reduce the vision sensor's marker-searching time, the initial searching position is obtained by the result of the block arrangement simulation. Once the vision sensor captures the positions of the 3 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 x-y plane is possible. The values of translation from the vision sensor to the base frame {B} can be obtained as

$$\begin{cases} x_{\{B\}} = x_{vision} + 300, \\ y_{\{B\}} = y_{vision} \end{cases} \quad (11)$$

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

$$M_{\{W\}} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} = {}^W_B T \begin{bmatrix} x_{\{B\}} \\ y_{\{B\}} \end{bmatrix} \quad (12)$$

where  $M_W$  is the marker position. The rotation values relative to the {W} frame are calculated using 3 different  $M_W$ s 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:

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

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 workpiece is always placed on the workplace. If  $d_z$  is not zero, the operator have to measure the height of the workpiece'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 genetic algorithm

Given that approximately 100~150 seams are randomly distributed on the workplace, the robot's traveling path should be minimal, resulting in the minimal working time. Among many optimization tools, the genetic algorithm (GA) is used for our robot system (Davidor, 1991; Fonseca & Flemming, 1995; Munasinghe et al, 2003). 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-effector is constant, the welding time can be calculated. But 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, the genetic algorithm that uses binary bit string encoding is prevalent. But, 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. Fig. 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.



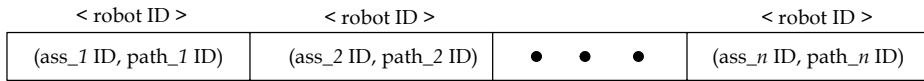


Fig. 15. Structure of the chromosome used for robot's path planning.

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 PMX (Partially Matched Crossover) suggested by Fonseca (Fonseca, 1995). 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 of the genetic information from the first parent by 2-points crossover and the other pieces are inherited according to the gene sequence of the second parent. The random number of each gene determines the possibility of crossover occurrence. 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  $P_1 = (1\ 2\ 3\ | \ 4\ 5\ 6\ 7\ | \ 8\ 9)$  and  $P_2 = (2\ 4\ 5\ | \ 3\ 7\ 6\ 9\ | \ 1\ 8)$ , 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 1st, 2nd, and 3rd paths are allocated to the first robot, and the 4th, 5th, 6th, and 7th paths are assigned to the second robot, and the 8th and 9th paths are assigned to the third robot. The '×' is an undefined gene. If the child inherits the second robot's genes,  $C_1 = (\times \times \times\ | \ 3\ 7\ 6\ 9\ | \ \times \times)$  and  $C_2 = (\times \times \times\ | \ 4\ 5\ 6\ 7\ | \ \times \times)$ , 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  $C_1 = (1\ 2\ \times\ | \ 3\ 7\ 6\ 9\ | \ 8\ \times)$  and  $C_2 = (2\ \times\ \times\ | \ 4\ 5\ 6\ 7\ | \ 1\ 8)$ . The final children are obtained as  $C_1 = (1\ 2\ 4\ | \ 3\ 7\ 6\ 9\ | \ 8\ 5)$  and  $C_2 = (2\ 3\ 9\ | \ 4\ 5\ 6\ 7\ | \ 1\ 8)$  by the mapping. Concurrently, the assigned genes for each robot are changed with respect to the cut points.

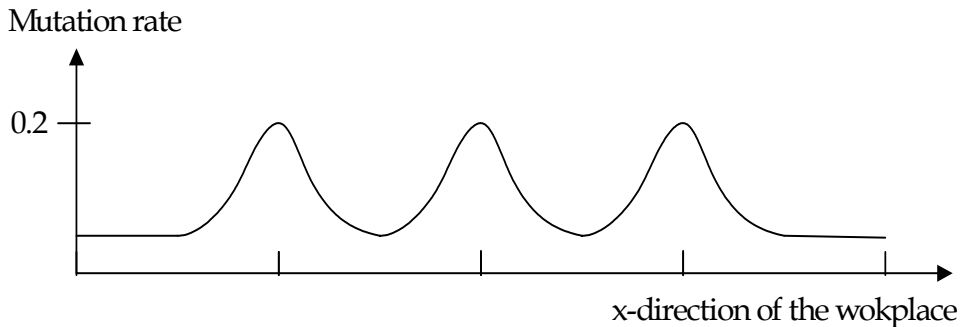


Fig. 16. The mutation rate of the spatially adaptive mutation algorithm.

Although the PMX is suitable to prevent a colliding 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. Fig. 16 shows the Gauss function.

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

where  $i = 0, \dots, n$ ,  $n$  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 follows:

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

where the range is the size of a population, and  $\text{drandom}()$  is a function that returns a random value. The position of the selected chromosome is defined as

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

where the base is the position of the last chromosome in a population. The selection constraints are assumed as follows: first, no robot can move beyond its working area restricted by the gantry crane's span; second, no collision is permitted; third, there is no limit of seam length for welding equipment. Examples of GA computing for a simple sub-assembly are shown in Fig. 17.

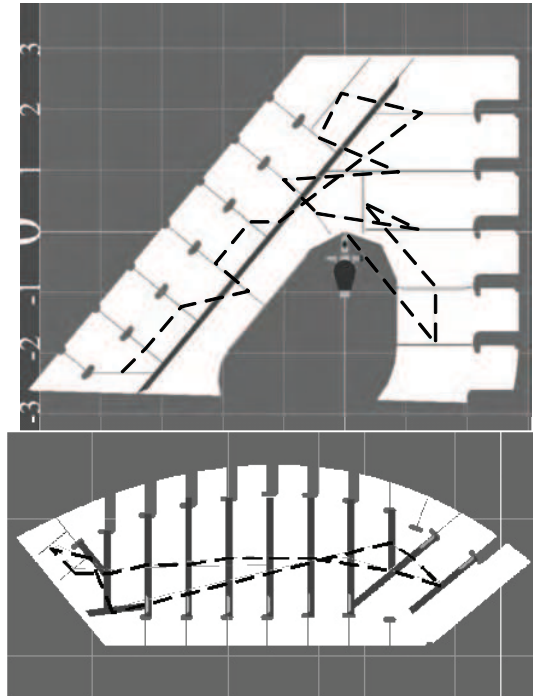
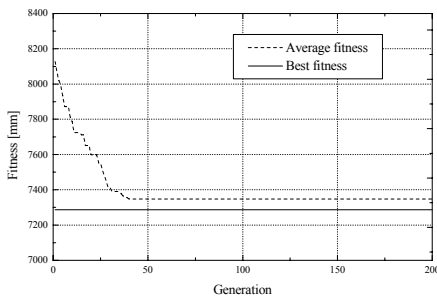
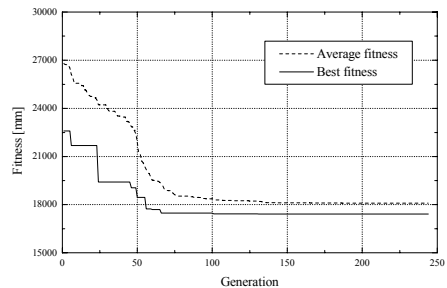


Fig. 17. Two examples of an optimal path generated for a single robot system: The dashed line indicates the optimal path represented by the robot base position.



(a) For the case of 3 robots.



(b) For the case of a single robot.

Fig. 18. Fitness of the GA algorithm for an example of Fig. 17. The best fitness value of (a) is 7,250 mm and the best fitness value of (b) is 17,000 mm. As we expected, the 3 robot system is more efficient than the single-robot system. In (a), we can see that due to the best chromosome generated during initial population computation, the best fitness value is constant over the entire generation.

Fig. 18 depicts the best fitness value and the average fitness value of each generation for 3 robots and for a single robot. For the multi robot system, to allocate workpiece to each robot, we engaged simple allotment method handled by operators. 2 main determinant factors of

allotments are welding length and collision avoidance. To prevent collision, we restricted robot's working volume by dividing workplace into the same numbered zone of robot. So the workpieces lying on each zone is 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 3 robots are more efficient than single robot according to the computation time and the best fitness value.

## 6. Implementation

The tools utilized in this work are as follows. Welding robots with the semi-automatic transfer system is applied to the grand-assembly and fully automatic robot system is applied to the sub-assembly. To implement the off-line programming both of grand- and sub-assembly: the PC of a Pentium IV 2.4 GHz processor with Windows 2000; Visual C++ for main programming and other functions; Open Inventor for graphic environment; TCP/IP for communication among the off-line programming PC, on-site monitoring PC and controller. Here, because the 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. The implemented video card has a 64 MB frame buffer memory and a 128 MB texture memory for our system. Also it is optimized to process Open GL. Also, vision system and mark-recognition algorithm is also developed to help welding robot system in shipbuilding.

On account of OOP(Object Oriented Programming), the user can selectively use each function of OLP, which automatically performs all functions in real time. Considering users' convenience, for the grand-, mid- and sub-assembly welding robot systems, robot simulation and robot program automatic generation functions were mainly used. And for the sub-assembly welding robot system, the whole operation of PC-based off-line programming is used. Table 1 shows the effectiveness of welding robot system with PC-based off-line programming.

Approximate time to	With developed OLP	On-line teaching
Generate a standard program	under 2 sec	About 20 min
Generate all standard programs	under 5 min	1 week
Generate job programs	under 5 min	1 hour
Total welding time for a general type workpiece (about 3m length, 3pass)	Under 5 min	About 30 min

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

## 7. Conclusion

This chapter described the welding robot system in a shipbuilding industry and the PC-based off-line programming system. The welding robot system is very helpful not only to increase productivity but also to solve worker's health problem. The developed off-line programming 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, developed off-line programming is highly compatible with other software, which fact allows the use of off-line programming on the Internet.

Thereby, adjustments to various robot systems are easy. Developed off-line programming is very easy for operators to use and maximizes the operating efficiency of welding robot systems of 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.

The contributions of this chapter are as follows: (1) a methodology for applying a welding robot system that works for variously shaped objects, especially for assembly lines of shipbuilding, is suggested; (2) the functions required to implement an OLP successfully and the development of a PC-based OLP that helps on-site operators handle a robot system easily are explained; and (3) 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 GA of robot path planning, are shown.

Using the techniques of welding robot system, utilizations for other hard process of shipbuilding such as painting and grinding are future works.

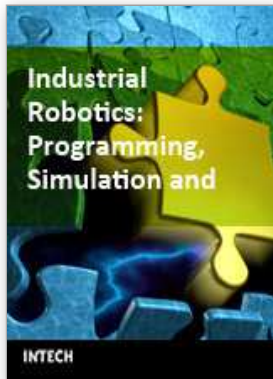
## 8. Acknowledgements

The research described in this chapter has been supported by the Research Center for Logistics Information Technology (LIT) designated by the Ministry of Education and Human Resources Development, Korea.

## 9. References

- Aiyama, Y. & Tanzawa, H. (2003). Management system for distributed and hierarchical robot groups, *Advanced Robotics*, Vol. 17, No. 1, pp. 3-20, ISSN: 1568-5535
- Bae, K. Y.; Lee, J. H. & Jung, C. W. (1998). A study on the development of an arc sensor and its interface system for a welding robot, *Journal of the Korean Welding Society*, Vol. 16, No. 3, pp. 129-140, ISSN: 1225-6153
- Borm, J. H. & Choi, J. C. (1992). Improvement of local position accuracy of robots for off-line programming, *KSME International Journal*, Vol. 6, No. 1, pp. 31-38, ISSN: 1011-8861
- Buchal, R. O.; Cherchas, D. B.; Sasami, F. & Duncan, J. P. (1989). Simulated off-line programming of welding robots, *International Journal of Robotics Research*, Vol. 9, No. 3, pp. 31-43, ISSN: 0278-3649
- Carvalho, G. C.; Siqueira, M. L. & Absi-Alfaro, S. C. (1998). Off-line programming of flexible welding manufacturing cells, *Journal of Materials Processing Technology*, Vol. 78, No. 1-3, pp. 24-28, ISSN: 0924-0136
- Choi, M. H. & Lee, W. W. (2003). Quantitative Evaluation of an Intuitive Teaching Method for Industrial Robot Using a Force / Moment Direction Sensor, *International Journal of Control, Automation, and Systems*, Vol. 1, No. 3 pp.395-400, ISSN: 1598-6446
- Craig, J. J. (1986). *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, ISBN: 0-201-09528-9, Reading, Massachusetts
- Davidor, Y. (1991). *Genetic Algorithms and Robotics: A Heuristics for Optimization*, World Scientific, ISBN: 9-810-20217-2, Singapore

- Ferretti, G.; Filippi, S.; Maffezzoni, C.; Magnani, G. & Rocco, P. (1999). Modular dynamic virtual-reality modeling of robotic systems, *IEEE Robotics and Automation Magazine*, Vol. 6, No. 4, pp. 13-23, ISSN: 1070-9932
- Fonseca, C. M. & Flemming, P. J. (1995). Genetic algorithms for multi objective optimization, *Evolutionary Computation*, Vol. 3, No. 1, pp. 1-16, ISSN: 1063-6560
- Goldberg, D. & Lingle, R. (1985). Alleles, Loci, and the TSP, *Proceedings of the First International Conference on Genetic Algorithms*, pp.154-159, ISBN: 0-805-80426-9, London, Oct., 1985, Lawrence Erlbaum Associates, Hillsdale, NJ
- Hoffmann, C. M. (1989). *Geometric and Solid Modeling: An Introduction (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*, Morgan Kaufmann, ISBN: 1-558-60067-1
- Kobayashi, N.; Ogawa, S. & Koibe, N. (2001). Off-line teaching system of a robot cell for steel pipe processing, *Advanced Robotics*, Vol. 12, No. 3, pp. 327-332, ISSN: 1568-5535
- Kreuzer, B. & Milojevic, D. (1998). Simulation tools improve planning and reliability of paint finishing lines, *Industrial Robot*, Vol. 25, No. 2, pp. 117-123, ISSN: 0143-991X
- Kunii, T. L. & Shinagawa, Y. (1992). *Modern Geometric Computing for Visualization*, Springer-Verlag, ISBN: 0-387-70105-2, Tokyo
- Munasinghe, S. R.; Nakamura, M.; Goto, S. & Kyura, N. (2003). Trajectory Planning for Industrial Robot Manipulators Considering Assigned Velocity and Allowance Under Joint Acceleration Limit, *International Journal of Control, Automation, and Systems*, Vol. 1, No. 1, pp.68-45, ISSN: 1598-6446
- Nagao, Y.; Urabe, H.; Honda, F. & Kawabata, J. (2000). 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, ISSN: 1568-5535
- Nielsen, L. F.; Trostmann, S.; Trostmann, E. & Conrad, F. (1992). Robot off-line programming and simulation as a true CIME-subsystem, *Proceedings of 1992 IEEE International Conference on Robotics and Automation*, pp. 1089-1094, ISBN: 0-818-62720-4, Nice, France, May, 1992, IEEE Computer Society Press
- Okumoto, Y. (1997). Advanced welding robot system to ship hul assembly, *Journal of Ship Production*, Vol. 13, No. 2, pp. 101-110, ISSN: 1542-0469
- Richard, G. (1997). Object-oriented programming for robotic manipulator simulation, *IEEE Robotics and Automation Magazine*, Vol. 4, No. 3, pp. 21-29, ISSN: 1070-9932
- Sheu, P. C-Y. & Xue, Q. (1993). *Intelligent Robotic Planning Systems*, World Scientific, ISBN: 9-810-207558-1, Singapore
- Wernecke, J. (1994). *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor*, Addison-Wesley, ISBN: 0-201-62495-8, Canada



## **Industrial Robotics: Programming, Simulation and Applications**

Edited by Low Kin Huat

ISBN 3-86611-286-6

Hard cover, 702 pages

**Publisher** Pro Literatur Verlag, Germany / ARS, Austria

**Published online** 01, December, 2006

**Published in print edition** December, 2006

This book covers a wide range of topics relating to advanced industrial robotics, sensors and automation technologies. Although being highly technical and complex in nature, the papers presented in this book represent some of the latest cutting edge technologies and advancements in industrial robotics technology. This book covers topics such as networking, properties of manipulators, forward and inverse robot arm kinematics, motion path-planning, machine vision and many other practical topics too numerous to list here. The authors and editor of this book wish to inspire people, especially young ones, to get involved with robotic and mechatronic engineering technology and to develop new and exciting practical applications, perhaps using the ideas and concepts presented herein.

### **How to reference**

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Chang-Sei Kim, Keum-Shik Hong and Yong-Sub Han (2006). Welding Robot Applications in Shipbuilding Industry: Off-Line Programming, Virtual Reality Simulation, and Open Architecture, Industrial Robotics: Programming, Simulation and Applications, Low Kin Huat (Ed.), ISBN: 3-86611-286-6, InTech, Available from: [http://www.intechopen.com/books/industrial\\_robotics\\_programming\\_simulation\\_and\\_applications/welding\\_robot\\_applications\\_in\\_shipbuilding\\_industry\\_off-line\\_programming\\_virtual\\_reality\\_simulatio](http://www.intechopen.com/books/industrial_robotics_programming_simulation_and_applications/welding_robot_applications_in_shipbuilding_industry_off-line_programming_virtual_reality_simulatio)

# **INTECH**

open science | open minds

### **InTech Europe**

University Campus STeP Ri  
Slavka Krautzeka 83/A  
51000 Rijeka, Croatia  
Phone: +385 (51) 770 447  
Fax: +385 (51) 686 166  
[www.intechopen.com](http://www.intechopen.com)

### **InTech China**

Unit 405, Office Block, Hotel Equatorial Shanghai  
No.65, Yan An Road (West), Shanghai, 200040, China  
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元  
Phone: +86-21-62489820  
Fax: +86-21-62489821

© 2006 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](#), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.