“Plug & Produce” Functions for an Easily Reconfigurable Robotic Assembly Cell

ID: AA-06-337

Revisions

- Section 1: “bent conveyor” → “belt conveyor”
- Caption of Figure 3: “System Holarchy” → “System Hierarchy”
- Section 2.2: Some comments on the full automation of the calibration procedure are added:

  > The procedure could be fully automated if we put numerous cameras around the assembly cell, but that would not pay in many cases because most of the cameras will be unused.

- Section 3.1: Reference for Dijkstra’s algorithm (Aho et al., 1983) added.
- Section 3.3: Some comments on the updating of calibration/adjacency graphs are added:

  > “that is, a new device node for the robot and its corresponding arcs are added to each graph.”

- Section 5: Some comments on our task allocation algorithm for the presented assembly system are added:

  > Our task allocation algorithm, which determines both a manipulator to assemble parts and routing of the parts, is based on inter-agent negotiation. The detail of the algorithm and simulation results can be found in (ARAI et al., 2003).

- Section 5: Some remarks on dealing with more complex assembly operations are added:

  > The presented assembly system is a prototype and can deal with only simple assembly operations. Complex operations, such as coordinated assembly by multiple manipulators, requires more advanced functions including online motion planning and three-dimensional workspace allocation, which future work should address.

Purpose

Aims to develop an easily reconfigurable assembly cell.

Design/methodology/approach

We implement some functions to resolve problems associated with physical reconfiguration of an agent-based robotic assembly cell, such as position calibration and workspace allocation.

Findings

The implemented prototype assembly cell is composed of industrial manipulators and a belt conveyor. Installation of a new manipulator and assembly execution is successfully demonstrated on the prototype cell.

Practical implications

In the developed assembly system, installation and removal of assembly devices are easily performed so that it can adapt to changes in the manufacturing environment quickly.

Originality/value

The developed system does not use specially designed hardware. We enable easy reconfiguration using conventional devices such as manipulators and belt conveyors.

Keywords

Assembly, robots, multi-agent systems, reconfigurability

1 Introduction

In recent years, reconfigurability of manufacturing systems has been in great demand (Koren et al., 1999; Chen, 2001). Changes of system configuration at low cost will help the systems cope with both predictable and unpredictable fluctuations in production environment. For example, a manufacturing system can adapt to the increase of production quantity if an additional device can be easily plugged into the system (Figure 1).

In regard to assembly systems, many researchers proposed decentralized control architectures to achieve flexibility, including reconfigurability (Oliveira, 1994; Valckenaers et al., 1995; Rizzi et al., 1997; Basran et al., 1997; Fraile et al., 1999; Chirn and McFarlane, 2000). Such decentralized software architectures are very helpful to achieve reconfigurability, but that covers only a part of the purpose. In actual systems, physical problems of reconfiguration, such as conflict of workspaces, should be solved effectively. In order to achieve reconfigurability in actual assembly systems, some researchers developed special homogeneous hardware (TAMAKI et al., 1993; Kondoh et al., 1998; Hanai et al., 1999), which is generally expensive.

The authors proposed an agent-based control architecture for robotic assembly cells that consist of heterogeneous conventional devices, such as manipulators and belt conveyors as shown in Figure 2 (Arai et al., 2001; SUGI et al., 2003). The decentralized architecture has functions that support easy reconfiguration; information renewals associated with physical reconfiguration are automated, and conflict of workspaces of

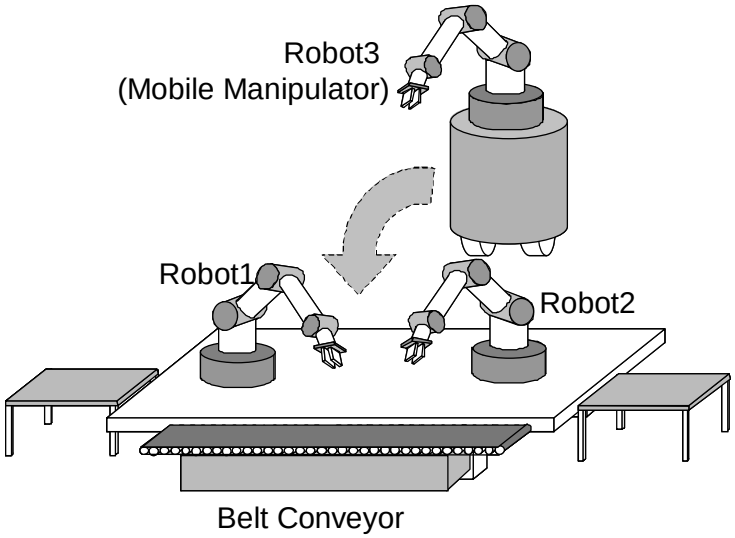


Figure 1: Plug-in of a New Robot

the devices at reconfiguration is solved decentralizedly by negotiation. Collectively, we call the functions "Plug & Produce" (Arai et al., 1997; SUGI et al., 2003).

As another physical problem, calibration of the position of a newly installed robot is required so that we can install the robot at an arbitrary location. Therefore, easy calibration is highly desirable. In our previous implementation (Arai et al., 2001), however, calibration is performed manually; that is, a human operator has to move the gripper of a newly installed robot to point at guiding marks to localize the robot. The necessity of manual calibration may spoil the system's reconfigurability.

In this paper, we present a new version of our assembly system, especially focusing on its new "Plug & Produce" functions, while the agent-based control architecture of the system (Figure 3) is described in (ARAI et al., 2003). The assembly system has a semi-automated calibration function for newly installed robots, which enhances the reconfigurability of the system significantly. The management mechanism of positional information resulting from installation/removal of robots is also described. Experimental results of the system's behavior in installation of a new robot and an assembly task are also shown.

2 Calibration for Plug & Produce

Calibration of the location of a newly installed robot is required for hand-over of parts and collision avoidance between the new robot and existing ones. We developed a semi-automated calibration method for Plug & Produce (Arai et al., 2002). In the method, relative position/orientation between a newly installed robot and an existing robot is easily obtained by observing markers attached to the end-effectors of the robots (Figure 4). Here we describe the method briefly.

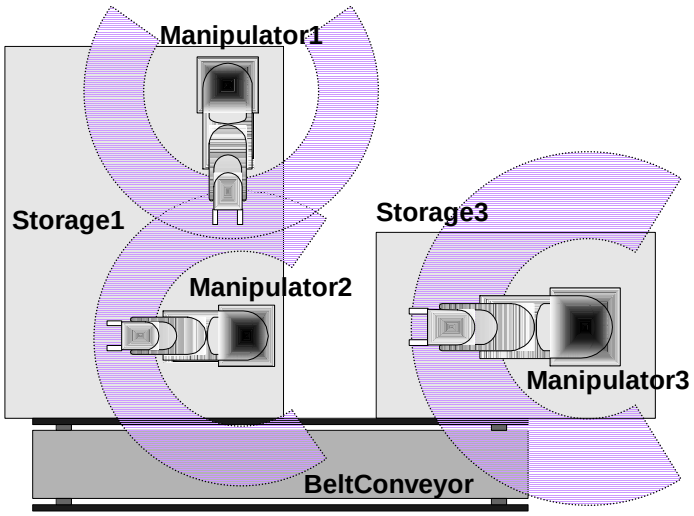


Figure 2: Robotic Assembly Cell (Example)

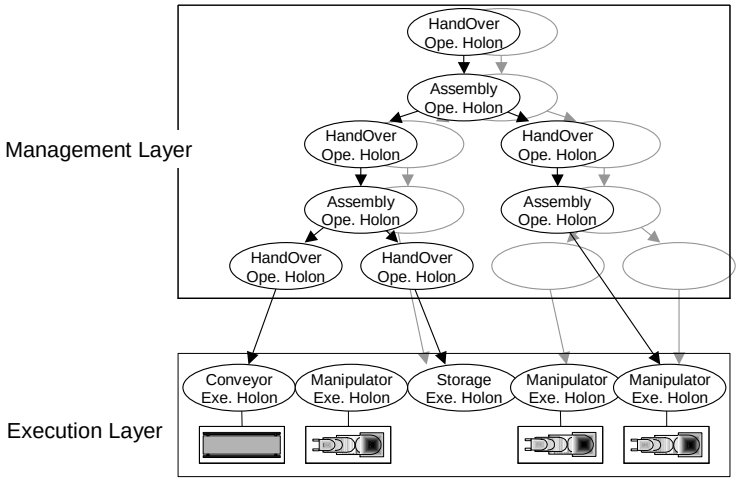


Figure 3: System Hierarchy

2.1 Direct Linear Transformation

Our calibration method is based on the reconstruction of 3D Cartesian coordinates of markers from camera images by the Direct Linear Transformation (DLT) method (Shapiro, 1978). Let us assume that two cameras observe a marker and we denote the image coordinates of the marker for Camera $i(= 1, 2)$ by (u_i, v_i). The DLT equations can be written as follows:

$$u_i = \frac{b_{i1}x + b_{i2}y + b_{i3}z + b_{i4}}{b_{i5}x + b_{i6}y + b_{i7}z + 1} \tag{1}$$

$$v_i = \frac{b_{i8}x + b_{i9}y + b_{i10}z + b_{i11}}{b_{i5}x + b_{i6}y + b_{i7}z + 1}, \tag{2}$$

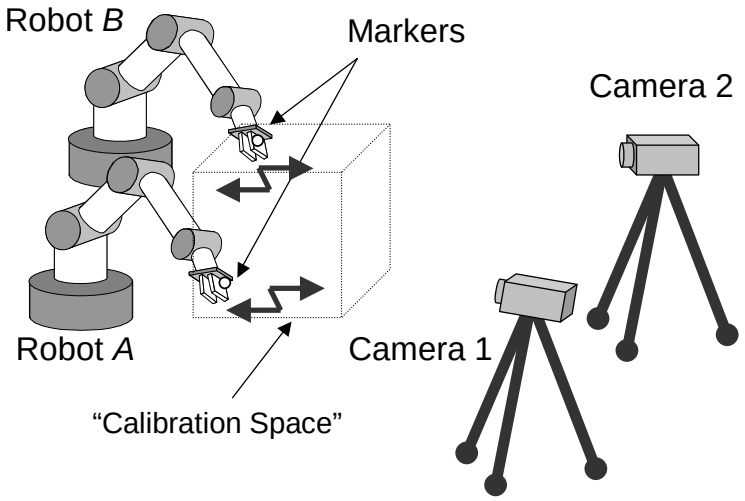


Figure 4: Calibration for Plug & Produce

where $\boldsymbol{x} = [x, y, z]^T$ is 3D Cartesian coordinates of the marker; $\boldsymbol{b}_i = [b_{i1}, \ldots, b_{i11}]^T$ are unknown DLT parameters. We can rewrite (1) and (2) in the following form:

$$\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{x}), \tag{3}$$

where $\boldsymbol{u} = [u_1, v_1, u_2, v_2]^T$. Once the DLT parameters are identified, 3D reconstruction as $\boldsymbol{x} = \boldsymbol{f}^{-1}(\boldsymbol{u})$ can be performed by the linear least-squares method.

2.2 Calibration Procedure

In our calibration method, we can obtain a homogeneous transformation matrix that represents positional relationship between the base frame of a newly installed robot and that of an existing one. The procedure for the calibration between robot *A* and *B* is as follows:

1. A human operator places two cameras so that they can observe the markers on robots *A* and *B*.

2. Robot *A* moves its end-effector autonomously in a limited space and the cameras observe the motion of its marker. Let us denote the image coordinates of the marker by Camera 1 and Camera 2 in the i-th observation by (u_{1Ai}, v_{1Ai}) and (u_{2Ai}, v_{2Ai}), respectively. The pairs of the image coordinates, $\boldsymbol{u}_{Ai} = [u_{1Ai}, v_{1Ai}, u_{2Ai}, v_{2Ai}]^T$, and 3D Cartesian coordinates of the marker in *A*'s base frame, ${}^A\boldsymbol{x}_{Ai} = [x_{Ai}, y_{Ai}, z_{Ai}]^T$, are sampled as control points $(i = 1, \ldots, n_A)$. Note that ${}^A\boldsymbol{x}_{Ai}$ can be calculated by *A*'s encoders.

3. Then robot *B* moves its end-effector autonomously in the limited space and the cameras also observe the motion of its marker. Let us denote the image coordinates of the marker by Camera 1 and Camera 2 in the j-th observation by (u_{1Bj}, v_{1Bj}) and (u_{2Bj}, v_{2Bj}), respectively. The pairs of the image coordinates, $\boldsymbol{u}_{Bj} = [u_{1Bj}, v_{1Bj}, u_{2Bj}, v_{2Bj}]^T$, and 3D Cartesian coordinates of the marker in *B*'s base frame, ${}^B\boldsymbol{x}_{Bj} = [x_{Bj}, y_{Bj}, z_{Bj}]^T$, are sampled as control points $(j = 1, \ldots, n_B)$. Note that ${}^B\boldsymbol{x}_{Bj}$ can be also calculated by *B*'s encoders.

4. A mutual positional relationship between the robots is calculated from $({}^{A}\boldsymbol{x}_{Ai}, \boldsymbol{u}_{Ai})$ and $({}^{B}\boldsymbol{x}_{Bj}, \boldsymbol{u}_{Bj})$, as a homogeneous transformation matrix $({}^{A}\boldsymbol{T}_{B})$.

In the last step, we can obtain ${}^{A}\boldsymbol{T}_{B}$ (and $\boldsymbol{b}_1$, $\boldsymbol{b}_2$) by iterative computation so that the following error index will be minimized (Arai et al., 2002):

$$J = \sum_{i=1}^{n_A} \left\| {}^{A}\boldsymbol{x}_{Ai} - \boldsymbol{f}^{-1}(\boldsymbol{u}_{Ai}) \right\| - \sum_{j=1}^{n_B} \left\| \boldsymbol{h}({}^{B}\boldsymbol{x}_{Bj}) - \boldsymbol{f}^{-1}(\boldsymbol{u}_{Bj}) \right\|, \tag{4}$$

where

$$\begin{bmatrix} {}^{A}\boldsymbol{x}_{Bj} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{h}({}^{B}\boldsymbol{x}_{Bj}) \\ 1 \end{bmatrix} = {}^{A}\boldsymbol{T}_{B} \begin{bmatrix} {}^{B}\boldsymbol{x}_{Bj} \\ 1 \end{bmatrix}. \tag{5}$$

The features of the calibration method are as follows:

- No substantial modification to assembly devices are required. All we need is markers on the robots and external cameras.
- Positions of cameras can be unknown. Thus we can place cameras instantaneously before calibration, and remove them immediately after calibration.
- The calibration procedure can be mostly automated. Human operators only have to place two cameras where they can observe the markers. The procedure could be fully automated if we put numerous cameras around the assembly cell, but that would not pay in many cases because most of the cameras will be unused.
- Transformation between the base frames of two manipulators is obtained. This is suitable for decentralized systems composed of autonomous agents.

In this method, calibration is made in a limited space (we call it "calibration space," Figure 4). Note that reliable accuracy is achieved only around the calibration space, where inter-robot operation (e.g., hand-over of parts) should be performed.

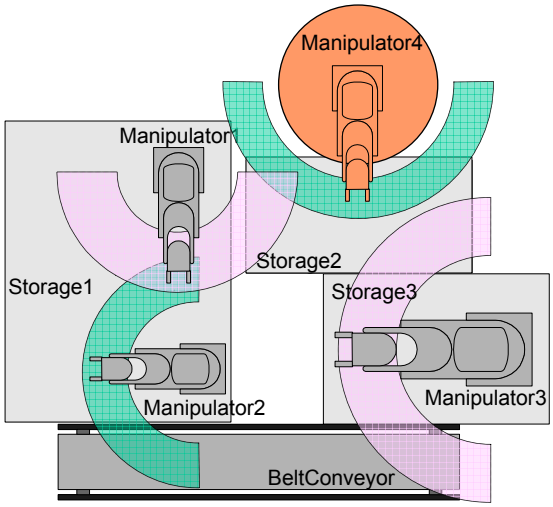
In our current implementation, the accuracy in the calibration space is about 0.5 [mm]. Details of the calibration including remarks on accuracy improvement are found in (Arai et al., 2002).

3 Management of Positional Information

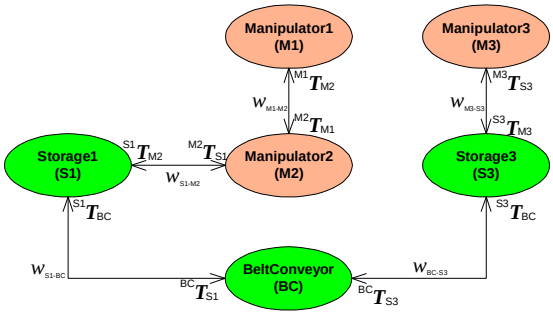
3.1 Graph Representation of Positional Relationships

Using the calibration method presented in the previous section, we can obtain information of positional relationships between robots. Our assembly system has a decentralized architecture, and therefore the positional information should be managed in a decentralized manner. Thus, we do not use global coordinates in a world coordinate frame, but just keep relative positional information between devices.

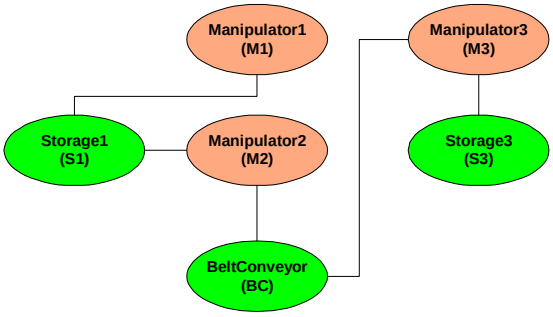
Let us consider an assembly cell like Figure 5(a). In this case, the positional information of assembly devices in the cell is represented as a "calibration graph," shown in Figure 5(b). Each node of the graph corresponds to each assembly device. Note that workbenches of the robots are also represented as "Storage" nodes. Only when the positional relationship between device *X* and *Y* is directly calibrated or defined, corresponding nodes are connected by an arc. In the arc, a homogeneous transformation matrix $({}^{X}\boldsymbol{T}_{Y})$ that represents the relative position/orientation between the devices



(a) Assembly Cell



(b) Calibration Graph



(c) Adjacency Graph

Figure 5: Graph Representation of Positional Relationships between Assembly Devices

is stored. By assigning a proper "cost" value, w_{XY}, to each arc according to the calibration accuracy, we can employ Dijkstra's algorithm (Aho et al., 1983) to find the "shortest" path between two devices that are not directly calibrated. The path gives relative position/orientation between the devices that are not directly calibrated.

An advantage of not using global coordinates is that the system does not have to resolve positional inconsistency explicitly. The following equation does not hold strictly

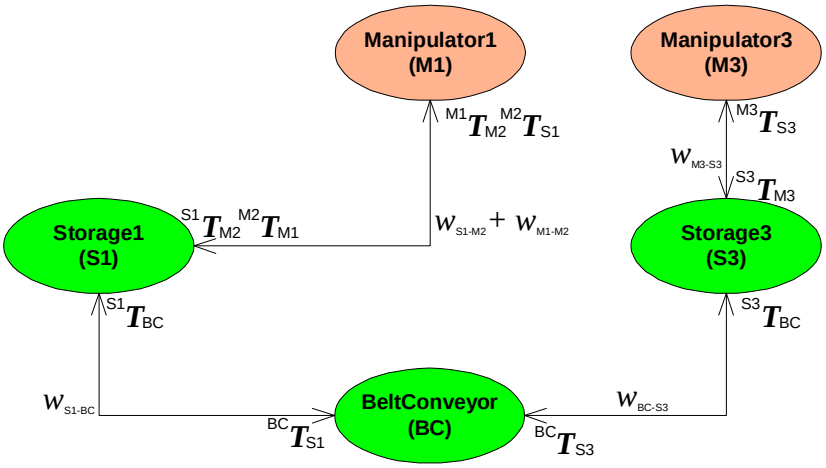


Figure 6: Updated Calibration Graph (Manipulator2 removed)

in our system because of the inevitable calibration errors:

$$ {}^{X}\boldsymbol{T}_{Z} = {}^{X}\boldsymbol{T}_{Y}\,{}^{Y}\boldsymbol{T}_{Z}. \tag{6} $$

Nevertheless, when using global coordinates, we have to determine each global position/orientation of the assembly devices as a compromise solution of inconsistent equations.

3.2 Workspace Allocation

After the positional information is obtained, proper workspace (re-)allocation is necessary to avoid physical conflict between assembly devices and to hand over parts and products. In our system, the workspaces of assembly devices are classified as follows (SUGI et al., 2003):

Shared Domain. A domain where the neighboring devices can enter. This domain is used for hand-over of parts and products.

Exclusive Domain. A domain where other devices cannot enter. This domain is used for assembly and keeping parts and products.

Shared and exclusive domains are defined as in Figure 7.

In our implementation, all the workspaces are represented as grids, and each grid point is labeled as shared, exclusive, or neither (Figure 8). When a robot is installed or removed, the information stored in the grid points must be updated according to the calibration result and movable areas of the robots.

3.3 Graph Representation of Adjacency

In addition to the calibration graph, the system has another graph, called “adjacency graph” (Figure 5(c)). In the adjacency graph, an arc connects two device nodes when the devices are neighbors (that is, they can hand over a part directly). Arcs in the adjacency graph do not connect two manipulators directly. This is because a manipulator

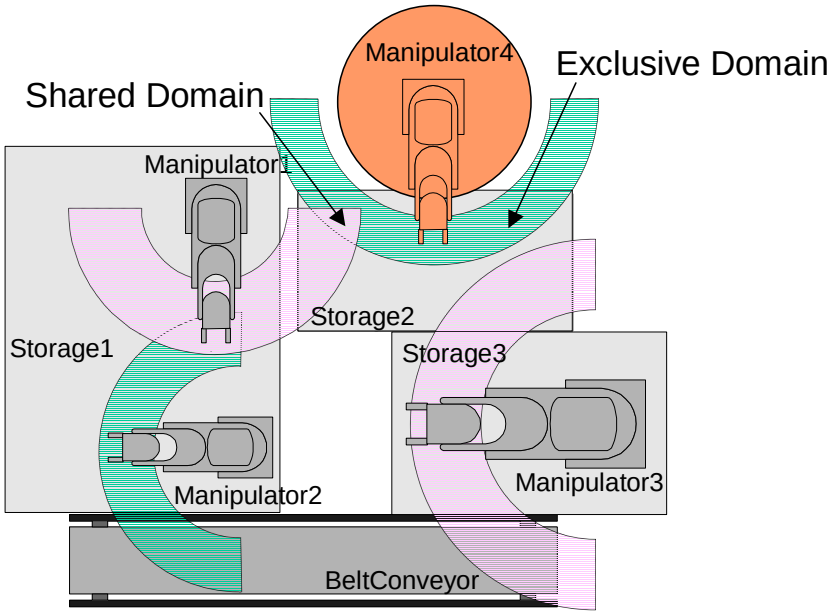


Figure 7: Shared/Exclusive Domains

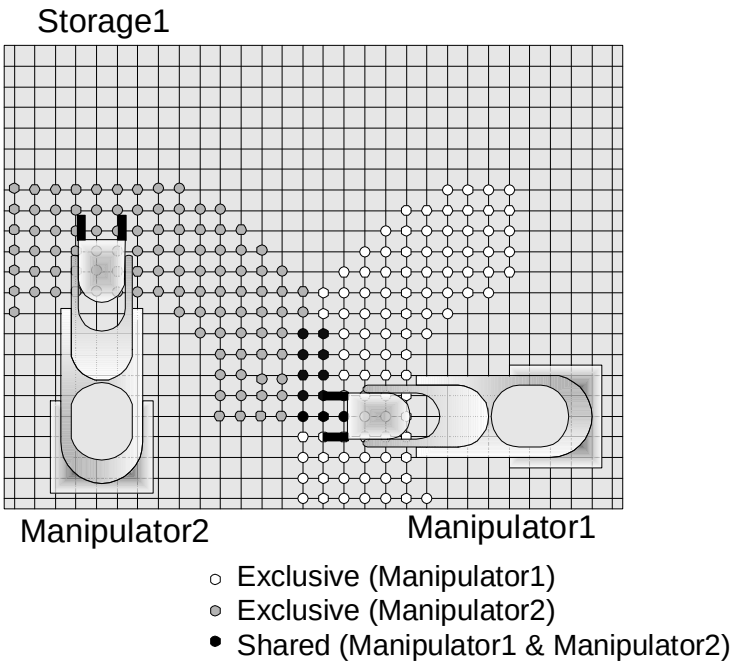


Figure 8: Grid Representation of Shared/Exclusive Domains

does not hand a part over to another manipulator directly, but places it on a shared domain temporarily and then another manipulator picks it up. The adjacency graph is required to determine transfer routes of parts between assembly devices (ARAI et al.,

2003).

When a new robot is installed in the system, both the calibration graph and the adjacency graph must be updated according to the calibration result; that is, a new device node for the robot and its corresponding arcs are added to each graph. Similarly, when a robot is removed from the system, both graphs must be updated, too. Note that, in the case of removal, the calibration graph must be kept connected. For example, when we remove "Manipulator2" from the assembly cell in Figure 5, the calibration graph is altered as Figure 6.

3.4 Procedure of Robot Installation and Removal

Now we describe how the positional information mentioned above is updated when a new robot is installed. The procedure of robot installation is as follows:

1. The new robot announces its rough position, which is given by a human operator, to all the devices and tells its possible neighbor devices to undertake no assembly operations temporarily for collision avoidance.
2. The new robot waits for the completion of all the assembly operations of its possible neighbors.
3. The calibration procedure described in Section 2 is performed. The procedure should be repeated for all the possible neighbors.
4. The calibration graph of the system is updated according to the calibration result.
5. Workspace information stored in the grid points on storages is updated according to the calibration graph and movable areas of the robots.
6. The adjacency graph is updated according to the change of workspace information.
7. Now, the robot installation is completed. The new robot tells the neighbor devices to resume undertaking assembly operations.

The procedure for removing a robot from the system is similar:

1. The robot to be removed stops to undertake new assembly operations and completes all of its assigned operations.
2. The robot hands over parts in its exclusive domain, if any, to other robots.
3. Workspace information stored in the grid points on storages is updated as a consequence of the removal of the robot.
4. The adjacency graph of the system is updated.
5. The calibration graph is updated.
6. Now, the removal of the robot is completed.

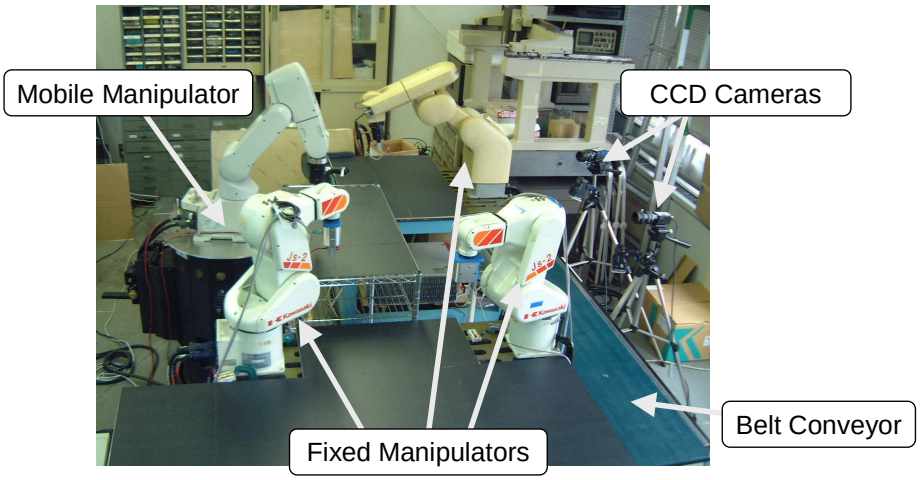


Figure 9: Implemented Assembly Cell

4 Experiment of Plug & Produce

4.1 Implemented Assembly Cell

Our implementation of the assembly system consists of three fixed manipulators, one mobile manipulator, and one belt conveyor (Figure 9). An LED is attached on the end-effector of each manipulator, as a marker for calibration described in Section 2. Two CCD cameras are also used for calibration. Each device is controlled by a Linux PC (Figure 10). All the assembly devices and storages are implemented as software agents. The agents are programmed in Java for high interoperability, while they were written in C++ in our previous implementation (Arai et al., 2001). We use Java Communications API and JNI (Java Native Interface) for controlling assembly devices. All the agents can communicate with each other through an Ethernet LAN.

4.2 Experiment

Installation of a mobile manipulator to the assembly cell was performed (Figure 11). First, the mobile manipulator ("Manipulator4") was placed next to a fixed manipulator ("Manipulator1"). Then each manipulator moved its gripper to eight points in turn ($n_A = n_B = 8$). These points formed a bounding box, whose location was determined based on the rough position of the newly installed Manipulator4. After the calibration procedure, a simple assembly task was requested. The task was just to collect part 'A' and 'B' and insert 'A' into 'B.' Necessary operations to complete the task were assigned to each device by negotiation and executed in order.

In the experiment, the new manipulator was successfully plugged into the system and participated in the assembly task immediately. Figure 12 is a Gantt chart for the experiment. First, calibration between Manipulator1 and Manipulator4 was executed. In our current implementation, the setting of tracking windows to observe markers for calibration is not yet automated, and therefore a human operator had to do it manually (60 [s]–105 [s]). Calibration was completed at 125 [s], and then an operation of moving part 'B' was immediately assigned to Manipulator4, the newly installed robot. At

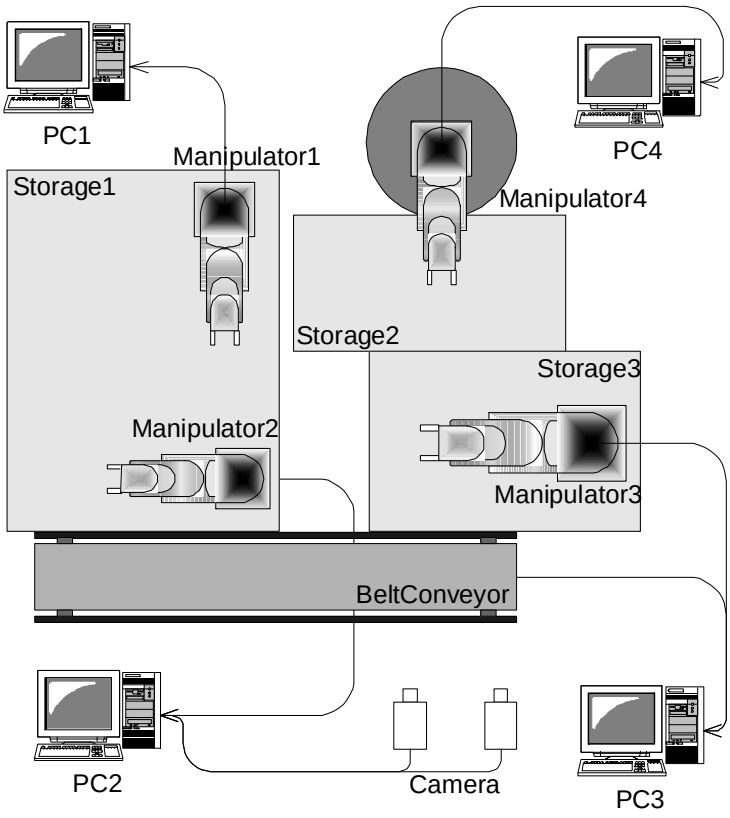


Figure 10: System Configuration

380 [s], the product was finally assembled.

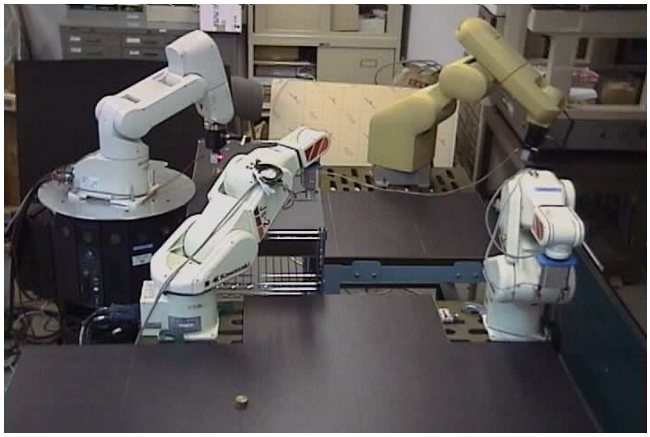
This experimental result shows that “Plug & Produce” functions work fine for the prototype assembly cell. Some other results on the execution of more complex assembly tasks by the cell can be found in (ARAI et al., 2003).

5 Conclusion

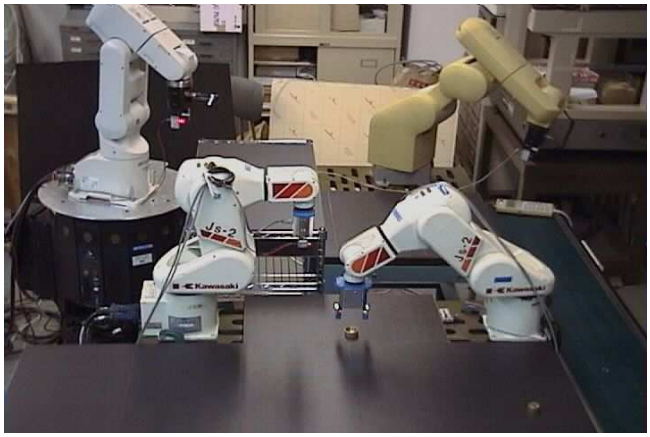
In this paper, we presented a robotic assembly system with high reconfigurability. The “Plug & Produce” functionality of the system including semi-automated calibration of locations of newly installed robots enables flexible reconfiguration of the system. In the experiment, we were able to install a new manipulator to the system easily to participate in assembly tasks.

The system will not be optimal with regard to its reconfigurability until it has effective task allocation. Our task allocation algorithm, which determines both a manipulator to assemble parts and routing of the parts, is based on inter-agent negotiation. The detail of the algorithm and simulation results can be found in (ARAI et al., 2003).

The presented assembly system is a prototype and can deal with only simple assembly operations. Complex operations, such as coordinated assembly by multiple manipulators, requires more advanced functions including online motion planning and three-dimensional workspace allocation, which future work should address.



(a) A Scene in Calibration



(b) A Scene in Assembly

Figure 11: Experiment of Plug & Produce

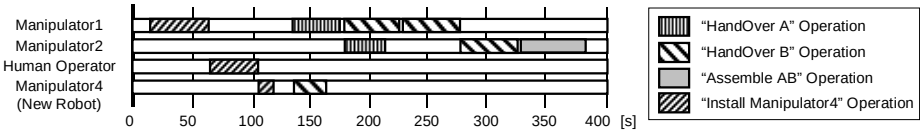


Figure 12: Gantt Chart for Plug-in and Assembly

Acknowledgments

This research is supported by IMS/HMS (Intelligent Manufacturing Systems/Holonic Manufacturing Systems) international research program.

References

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1983). *Data Structures and Algorithms*. Addison-Wesley.

Arai, T., Aiyama, Y., and Sasaki, Y. (1997). Holonic storage: An assembly and storage cell by manipulation using environment. In *Proc. of 29th CIRP Int. Seminar on Manufacturing Systems*, pages 221–226.

Arai, T., Aiyama, Y., Sugi, M., and Ota, J. (2001). Holonic assembly system with plug and produce. *Computers in Industry*, 46(3):289–299.

ARAI, T., IZAWA, H., MAEDA, Y., KIKUCHI, H., OGAWA, H., and SUGI, M. (2003). Real-time task decomposition and allocation for a multi-agent robotic assembly cell. In *Proc. of 5th IEEE Int. Symp. on Assembly and Task Planning*, pages 42–47, Besançon, France.

Arai, T., Maeda, Y., Kikuchi, H., and Sugi, M. (2002). Automated calibration of robot coordinates for reconfigurable assembly systems. *Annals of the CIRP*, 51(1):5–8.

Basran, J. S., Petriu, E. M., and Petriu, D. C. (1997). Flexible agent-based robotic assembly cell. In *Proc. of 1997 IEEE Int. Conf. on Robotics and Automation*, pages 3461–3466.

Chen, I.-M. (2001). Rapid response manufacturing through a rapidly reconfigurable robotic workcell. *Robotics and Computer Integrated Manufacturing*, 17(3):199–213.

Chirn, J.-L. and McFarlane, D. C. (2000). A component-based approach to the holonic control of a robot assembly cell. In *Proc. of 2000 IEEE Int. Conf. on Robotics and Automation*, pages 3701–3706.

Fraile, J.-C., Paredis, C. J. J., Wang, C.-H., and Khosla, P. K. (1999). Agent-based planning and control of a multi-manipulator assembly system. In *Proc. of 1999 IEEE Int. Conf. on Robotics and Automation*, pages 1219–1225.

Hanai, M., Nakasai, T., Hibi, H., Kojima, F., and Fukuda, Y. (1999). New autonomous manufacturing system adapted for uncertainty in market —APS: Adaptive production system—. In *Proc. of the Second Int. Workshop on Intelligent Manufacturing Systems*, pages 15–22.

Kondoh, S., Umeda, Y., and Yoshikawa, H. (1998). Development of upgradable cellular machines for environmentally conscious products. *Annals of the CIRP*, 47(1):381–384.

Koren, Y., Heisei, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G., and Brussel, H. V. (1999). Reconfigurable manufacturing systems. *Annals of the CIRP*, 48(2):527–540.

Oliveira, E. (1994). Cooperative multi-agent system for an assembly robotics cell. *Robotics and Computer-Integrated Manufacturing*, 11(4):311–317.

Rizzi, A. A., Gowdy, J., and Hollis, R. L. (1997). Agile assembly architecture: An agent based approach to modular precision assembly systems. In *Proc. of 1997 IEEE Int. Conf. on Robotics and Automation*, pages 1511–1516.

Shapiro, R. (1978). Direct linear transformation method for three-dimensional cinematography. *Research Quarterly*, 49(2):197–205.

SUGI, M., MAEDA, Y., AIYAMA, Y., HARADA, T., and ARAI, T. (2003). A holonic architecture for easy reconfiguration of robotic assembly systems. *IEEE Trans. on Robotics and Automation*, 19(3):457–464.

TAMAKI, K., UNO, M., MITA, T., SUGIMOTO, K., SAKAUE, S., and ONARI, H. (1993). A restructurable assembly center employing mobile dd-robots. In *Proc. of 24th Int. Symp. on Industrial Robots*, pages 119–126.

Valckenaers, P., Brussel, H. V., Bongaerts, L., and Bonneville, F. (1995). Programming, scheduling, and control of flexible assembly systems. *Computers in Industry*, 26(3):209–218.