Implementation of Integrated Cooperative Robots System for Tight Cooperation

Toyomi Fujita and Hiroshi Kimura
Graduate School of Information Systems
University of Electro-Communications
1-5-1 Chofu-ga-oka, Chofu, Tokyo 182-8585, JAPAN
(E-mail : fujita@hi.is.uec.ac.jp)

Abstract

The purpose of this research is to realize a multiple robots system which can be easily developed and managed. In order to perform robust cooperative work, it is necessary that multiple robots can achieve not only a high level of cooperation without communication but also simple cooperation with communication, and then switch between modes according to the situation. We define such kind of cooperation as “Tight Cooperation”. As for the system structure for the Tight Cooperation, we insist that there must be an integrated concept by which both the system of a robot and the entire system can be handled with same concept. Based on this consideration, an object oriented multiple robots system named “ICRoS” (Integrated Cooperative Robot System) is implemented to two hexapod robots with parallel processors. The effectiveness of that system is verified by four kinds of experiments for the cooperative lifting of a box as Tight Cooperation.

1. Introduction

Cooperative work by multiple mobile robots is effective for dangerous work at building sites, for repair work in stricken areas, etc. Therefore, the control system must be considered not only for cooperation with geometric exclusion such as collision avoidance[1], but also for cooperation with dynamical interaction between robots. Though there have been many studies of cooperation with dynamical interaction[2]-[7], this cooperation has been achieved based on a system by which multiple robots are controlled autonomously through the use of force information and the robots themselves hardly communicate with each other. However, it is difficult for the robot to deal with a situation autonomously in every case because the robot may not be able to predict situations which may change with the time. In unfamiliar situations, a comparatively simple cooperation, such as the cooperation in which robots are controlled statically using only position information with explicit communications between robots, become effective. In addition, if such a simple cooperation and the more advanced control using force information can be switched according to the situation, the level of robustness of work can be raised. We define this kind of cooperation with dynamic interaction in which suitable control method must be selected and switched as “Tight Cooperation”. The concept is shown in Fig.1.

For such Tight Cooperation, we have proposed a method for constructing a multiple robots system which can be easily developed and managed [8]. As for the system structure for the Tight Cooperation, we have insisted there must be an integrated concept by which both the system of a robot and the entire system can be handled with same concept. Based on this consideration, a cooperative working system, which is named “ICRoS” (Integrated Cooperative Robot System), is implemented to two hexapod robots. Then, four different experiments in which those robots lift a box cooperatively are carried out as an example of Tight Cooperation, and the effectiveness of ICRoS is verified.

2. Software Development Model

We considered utilizing the concept of Smalltalk as the software development model because the following basic characteristics are convenient in realizing a multiple robots system for Tight Cooperation.

- It is a typical object-oriented language, and it is possible to treat individual data in the system by an integrated concept as an Object.
- The description style of execution in the Smalltalk system is integrated as the message-passing.
- The execution mechanism of message-passing represents the communication between the Objects as it is.
- The procedure itself, which is called Block, and the characteristics of the execution environment such as
Process, Stack, etc. are also represented as an Object and can be managed.

- Many useful tools such as Inspector, Debugger, etc. are realized by the description style of Smalltalk by using the function of 2.

These coherent characteristics of Smalltalk are effective to realize an integrated concept by which both the system of a robot and the entire system can be handled with same concept. However, the virtual machine of original Smalltalk-80 (ST-80) has hardly considered the parallel execution of multiple processes and execution in a multiple processors environment. We therefore improved the virtual machine of ST-80 and developed a new Smalltalk for concurrent and parallel execution in a multiple processor environment[9]. This Smalltalk was named “ST-MPP” (SmallTalk for Multiple Processes and Processors), and the virtual machine was named “ST-MPP-VM”.

3. Multiprocess Execution

Though the description language of ST-MPP is the same as ST-80, the execution mechanism for multiple processes is different. In the ST-80, there is only one bytecode interpreter in one virtual machine. Thus, multiple processes are executed by the bytecode interpreter. On the other hand, in the ST-MPP, a process and a bytecode interpreter are represented as an Object ‘ST-MPP-Process’ and ‘B-Interpreter’ respectively. A B-Interpreter corresponding to one ST-MPP-Process is assigned when multiple processes are executed in ST-MPP, then each B-Interpreter is executed independently by utilizing the function of the kernel of the processor. An Object ‘CommunicationManager’ is also implemented to manage communication between B-Interpreters.

The multi-processor environment is realized with this system by executing an ST-MPP-VM in each processor. For example, when a robot is equipped with more than one processor, the number of ST-MPP-VMs executed equals the number of processors, as shown in Fig.2. By this mechanism, additional ST-MPP-VMs can be added as necessary inside the robot; thus, it has the flexibility to accommodate changes in hardware composition.

In addition, when a message is sent to an Object that has been generated in another ST-MPP-VM, the execution is performed in the ST-MPP-VM that generated the object. For example, in Fig.3, when a message is sent in a B-Interpreter that is executed in the ST-MPP-VM of VM1 to an Object generated by VM2 (1. in Fig.3), the contents of the message are sent to VM2 through the CommunicationManager of VM1 (2. in Fig.3). The CommunicationManager of VM2 generates a new B-Interpreter in that ST-MPP-VM, and the message is executed in the new B-Interpreter (3. in Fig.3). Then, once that execution is finished, the returned Object is sent to the VM1 that initiated the message, and the B-Interpreter of VM1 continues the execution normally (4. in Fig.3).

This management is performed within the ST-MPP-VM, and the form of access to the Object in the ST-MPP by user is the same regardless which ST-MPP-VM the Object belongs to. Therefore, the form of execution can be integrated whether the user is accessing an Object between robots, between processors, or inside in a processor.

4. Mechanism of Cooperation

As the next step, necessary functions for multiple robots were implemented on this ST-MPP. We named this multiple robots system “ICRoS” (Integrated Cooperative Robot System). In ICRoS, the tasks for cooperation can be represented as an Object of ST-MPP. [10] has represented a unit of the motion for the robot as an ‘Operation’ and has hierarchized the concurrent motion and the sequential series with the units of ‘ParOperation’ and ‘SeqOperation’, respectively. This concept is utilized in ICRoS, too. Operation, ParOperation, and SeqOperation are defined as an Object of ST-MPP respectively, then the cooperative work by multiple robots is represented by these Objects.
As an example, the mechanism for the case of cooperation in which two robots each grasp a box with two hands and then lift it up vertically is shown in Fig.4. When this cooperation is started, the following ST-MPP message is executed first.

- **Robot1** coopLift: aBox up: height with: **Robot2**

It can be considered that this task is represented as cooperated execution of the sub task for the lifting motion of each robot. Moreover, it can be considered that each task of lifting is represented as cooperated execution of the sub task for the up motion of each working arm. Then, these cooperated sub tasks are executed concurrently and synchronized. Therefore, the top task is represented as a ParOperation of ‘coopLift:up:with:’. In that ParOperation, two ParOperations for sub task of ‘liftUp:’ of each robot are registered. In these ParOperations, two Operations for sub task of ‘up’ of each working arm are registered respectively.

A ParOperation or Operation is executed by each corresponding ST-MPP-Process. Each ParOperation starts registered ParOperations or Operations for sub task concurrently, and waits for the finish of all sub tasks. When the execution of the sub ParOperation or Operation is finished, the ending is notified to the upper ParOperation and the execution is synchronized. This mechanism is realized by the message-passing of ST-MPP. In addition, these executions are performed by the same method whether the communication between robots is performed or not. For example, in Fig.4, when the top ParOperation starts sub ParOperations of ‘liftUp:’, the execution causes the communication from Robot1 to Robot2 because the top ParOperation is executed in Robot1. Although ParOperations of ‘Robot1 liftUp:’ is started by same method, the communication between robots is not performed.

As in this example, cooperation between multiple robots can be represented hierarchically as an Object, and the execution mechanism is integrated whether the cooperation is between robots or in an individual robot. This indicates that the software can be developed by integrated concept regardless of the number or the location of processor in ICRoS. Other kinds of cooperative work can also be widely realized by the same mechanism.

### 5. Management Tools

In ICRoS, the following Objects for management tool have been implemented.

- ‘MultiProcessManager’ by which multiple parallel processes can be managed and debugged.
- ‘CooperationManager’ by which the cooperation can be managed.
- ‘RemoteSendingBrowser’ by which the communication between ST-MPPs can be analyzed.

These tools are very useful in the debugging, analysis, and improvement of work.

---

**Figure 4:** example of cooperative lifting work.

**Figure 5:** overview of CooperationManager.

---

Fig.5 shows the CooperationManager for the cooperative work of Fig.4. CooperationManager is started when some error in execution occurs or the user explicitly interrupts the execution. Then, after each of the processes of all tasks for cooperation are stopped, the hierarchical structure of ParOperations and Operations which are associated with those tasks are displayed. In Fig.5, each line indicates each ParOperation and Operation of Fig.4. The color is reversed on the selected line, and the Object for the process corresponding to the selection is displayed on the screen below. The execution can be analyzed by starting the debugger for the process. Then, the relation of processes for cooperation can be analyzed by similarly starting the debuggers for the other processes and by using MultiProcessManager.

Fig.6 shows the RemoteSendingBrowser which is useful to investigate the communication between ST-MPPs. In RemoteSendingBrowser, the list of message-passing which caused the communication between ST-MPPs in one execution is displayed with the number and the channel. The user can analyze each message-passing by using the RemoteSendingBrowser.

In ICRoS, some useful management tools for multiple robots can be realized easily because every element in the
system is represented as an Object of ST-MPP, and the accessibility is ensured. This characteristic is an important advantage which has not been available in conventional robot control system.

6. Experiments

6.1. Experimental System

ICRoS was implemented to two hexapod robots having the same hardware composition. Fig.7 indicates the overview of the experimental system. ST-MPP was implemented on Transputer T9000, a parallel processor. One T9000 is attached to each robot, and the ST-MPP is executed on each T9000. A user interface can be performed at the terminal computer. A window for each ST-MPP can be shown on the display, then the operation shown in each window is associated with the ST-MPP corresponding to that window.

In order to verify the effectiveness of this system, four kinds of experiments by which two hexapod robots lift a box cooperatively were carried out. In these experiments, a hexapod robot supports itself on four legs, grasps a box using two fore legs as a working arm, then lifts a box up cooperatively with another robot. This cooperation is performed according to the method shown in Fig.4. In this work, it is assumed that the handled box is a rectangular parallelepiped shape and that it made of a hard material such as wood so that the transformation occurs due to contacting is negligible. It is also assumed that this work is performed statically.

This cooperative work was performed under the following two different kinds of control.

1. Control using only position information.
2. Control using force information.

Three kinds of experiments for 1. were carried out, and one experiment for 2. was carried out. Each details are described in Section 6.2 and 6.3 respectively.

6.2. Cooperation using position control only

In this kind of cooperation, the tiptoe position of each working arm is controlled so that relative deflection with the tiptoe of other working arms remains constant in order to keep the target box horizontal. This cooperation is shown in Fig.8. The following equation indicates the desired tiptoe position vector $P_{t_i}$ of each working arm-i when the desired position vector of the box is $P_{box}$, at the working time t:

$$
\begin{align}
P_{t_i} - P_{box} &= P_{t_i} - P_{box} \\
(i = 0) \\
P_{t_i} - P_{box} &= P_{box} - P_{box} \\
(i = 1, 2, 3)
\end{align}
$$

Here, $P_{t_i}$ and $P_{box}$ indicate the position vector of the tiptoe of arm-i and the box at the start time respectively. The following three kinds of experiments for this kind of cooperation were carried out.
Experiment-A
The goal height of lifting is subdivided, ParOperations for each ‘coopLift:up:with:’ are set, and they are registered in a SeqOperation. These ParOperations are executed sequentially when the SeqOperation begins. At the beginning and end of these ParOperation executions, four working arms are synchronized, and the desired position of each tiptoe is adjusted every time.

As the result, prototyping of the cooperation between the robots could be easily achieved because of the functions of ICROs based on integrated concept although the amount of communication was large. In addition, the debugging could be easily performed using CooperationManager shown in Fig.5.

Experiment-B
The method of cooperation is the same as A. However, the Objects which represent the numeric information and whose content don’t change are defined as ‘ReadOnlyObject’. If a message is sent to the ReadOnlyObject which was generated in the other ST-MPP, then the content is copied from the owner ST-MPP-VM and the Object becomes a local Object in the ST-MPP which sent the message. Afterwards, access to the other ST-MPP is not occurred when the same message is sent again.

In this experiment, an improvement of the software could be achieved. The registration of ReadOnlyObject could be easily performed by using RemoteSendingBrowser shown in Fig.6.

Experiment-C
In this experiment, event-driven execution is performed. An Object ‘EventManager’ is implemented to manage event-driven processing. The EventManager observes relative deflection at the tiptoe position of each working arm and generates the event when the work enters an improper state. In contrast with Experiment-A and B, the goal height of lifting isn’t subdivided in this experiment, and only one ParOperation for ‘coopLift:up:with:’ is executed.

As the result, the event-driven processing could be implemented easily and the quality of the control could be improved because the accessibility to the data is ensured as an Object for the execution environment such as Process, Stack, etc.

Table 1 shows the results for the amount of communication between robots and the execution time for Experiment-A, B, and C. The ratio to compare the results to those of Experiment-A is shown.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount of Communication (%)</td>
<td>100</td>
<td>24.3</td>
<td>9.1</td>
</tr>
<tr>
<td>Execution time (%)</td>
<td>100</td>
<td>37.5</td>
<td>16.1</td>
</tr>
</tbody>
</table>

- The execution mechanism includes the communication itself. Therefore, the communication between robots does not have to be specified, and the software for cooperation of multiple robots can be developed using the same concept as that for a single robot.
- The accessibility to the data is ensured in the system. Therefore, some useful tools can also be developed easily according to the purpose.

6.3. Cooperation using force information
In this experiment, the contact force to satisfy the necessary stability of each working arm is computed first, then each tiptoe’s contact force is adjusted to maintain the stability by using the information for each tiptoe reaction force. As for the method of computing the contact force of the working arm, [3],[4],[7], etc. which considers the dynamic problem of cooperation are referred to. Here it is assumed that the target box is horizontal, and the equivalence of force and moment for the horizontal face is considered as shown in Fig.9.

When $F_i (i = 0, 1, 2, 3)$ is a vector of the force which acts on the target at each contact point, the equation can be written as follows:

$$\sum_{i=0}^{3} F_i = 0 \quad (2)$$

The equivalence for the moment is considered next. Suppose that two two-dimensional vectors $A = [a_0 b_0]^T$ and $B = [c_0 d_0]^T$ are given and $\otimes$ is the operator by which $A \otimes B = ad - bc$ is performed. Let $P_n$ be the position vector at the contact point $P_n$ of the arm-$n$. Then, the equation of the equivalence for the moment around $P_n$ is indicated as follows:

$$\sum_{i=0}^{3} (P_i - P_n) \otimes F_i = 0 \quad (n = 0, 1, 2, 3) \quad (3)$$

Each contact force in horizontal face is computed so that these equations are satisfied and there is no sliding at the contact point. In regard to the direction of the vertical axis, it is assumed that the influence of the moment can be disregarded, and only the equivalence of force is considered. The EventManager observes the reaction force of each working arm and generates the event when the state become unstable. An overview of this cooperative work is shown in Fig.10. This execution is also started by one ParOperation.
for ‘coopLift:up:with:’ as well as Experiment-C explained in Section 6.2.

In this experiment, the cooperation could be easily achieved because the improvements from Experiment-C were implemented by the integrated concept of ST-MPP without the change of the execution mechanism of software. Therefore, it can be considered that ICRoS is very effective for Tight Cooperation because the cooperation by different control methods can be realized easily.

7. Conclusion

In cooperative work with dynamic handling such as transportation, the situation of the work may change over time and become critical. Based on this consideration, we insisted that cooperative work by which the control method can be switched is effective. We defined such cooperative work “Tight Cooperation”. We implemented a cooperative working system “ICRoS” based on an appropriate method for the construction of multiple robots system for the Tight Cooperation. “ST-MPP” was implemented as the software development model based on an object oriented concept, and some fundamental functions for multiple robots were developed on the ST-MPP. In addition, four kinds of experiments were carried out under the two different kinds of control. Through these experiments, we verified that ICRoS was useful on the point of development, management, and expandability for Tight Cooperation by multiple robots.

References


