Robot Middleware Architecture Mediating Familiarity-Oriented and Environment-Oriented Behaviors

Akihiro Kobayashi, Yasuyuki Kono, Atsushi Ueno, Izuru Kume, Masatsugu Kidode
{*akihi-ko, kono, ueno, kume, kidode*}*@is.aist-nara.ac.jp*
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma 630-0192, Japan

Abstract

This paper presents a middleware architecture for personal robots applied to various environments. The architecture allows a robot to consistently integrate environment-oriented applications with its original and familiar characteristics for its user. The familiar characteristics and environment-oriented applications tend to be developed independently. However, the two kinds of functions should share sensors and actuators to generate consistent actions. To this end, we have analyzed the relationship between robot actions and their mental effects on the user, and have designed middleware, called the "mediator," to play the role of mediator by dynamically selecting either: 1) sequential execution, 2) time-sharing execution, or 3) concurrent execution. We had an experiment to simulate the mediation, and to estimate the familiarity and efficiency.

1 Introduction

In this paper, we propose a framework to give personal robots the ability to supply local, specific services in each environment while keeping their original familiarity. Recent research provides two viewpoints of the usage of autonomous mobile robots. On one hand, a robot is thought of as a mobile and intelligent interface to information systems [1, 10, 7]. On the other hand, robot owners expect their robots to behave similar to a familiar and amusing pet. Therefore, human-like expression on their appearance and motion is an important issue [4, 5].

In the near future, a personal robot, which accompanies its owner, should gain the ability to interface with an information system in an office, as long as the robot stays in the office. This ability to interface gives benefits to both the owner of the robot and the service provider of the office. From the owner's point of view, accessing the system through his or her familiar robot is desired. The information system can use the personal robot as its physical interface for a visitor, although a traditional service provider needs robots to provide these services. The benefits are summarized in the following, and examples are shown in Fig. 1

- Robot Owner
  - The owner is provided with services in each environment.
  - The owner can use the familiar interface by using familiar hardware.
  - Adopted behaviors prevent the owner from getting bored with his/her robot.
  - More than one robot in an environment must work collaboratively in the event of any collisions.
- Service Provider at an Environment
  - It is not necessary to stock many robots to provide service, e.g., for guiding visitors.
  - The robot should follow TPO rules without effort.

Guide for Unknown Building
EXIT
Keep Quiet in a Library
Collaborate with other Intelligent Systems

Figure 1: New Abilities of Personal Robots Realized by the Middleware Architecture

A robot should execute simultaneously both the originally-programmed behaviors and the newly-loaded behaviors, for familiar and useful actions. For example, the robot behaves as a pet in its interaction with the owner, while the owner is accompanied by his/her robot to their destination. However, these two roles tend to interfere with each other in the action of the robot. Generally, parallel resource management architecture is needed to resolve resource-interference among independently developed applications. A new theme in robotics is the integration of two software components, with both components never knowing how their opponent manages common resources, i.e., the physical devices in the robot.

2 Mediation

2.1 Mediation Framework

A new framework for mediation is needed for a robot to execute multi-applications in parallel, but developed independently. Robot programmers have traditionally developed whole applications in the robot on the condition that unknown applications will never be executed. Controlling the robot's motion carefully is essential because a robot has access to the real world. Nonetheless, keeping a robot's user and the robot secure while independent applications run on it in parallel is quite difficult.

In order to cope with dynamic coordination, we propose a new process model based on *mediation*, instead of coordination. We assume the middleware on which all applications run. The middleware provides applications with basic functions for a mobile robot and mediates between any access to devices from its applications. In traditional robot programming, multi applications are coordinated when they are designed. The proposed middleware is an online interpreter that can block competitive device accesses, and can schedule threads dynamically using only behaviors and additional information. We call this middleware a *mediator*. (Fig. 2)

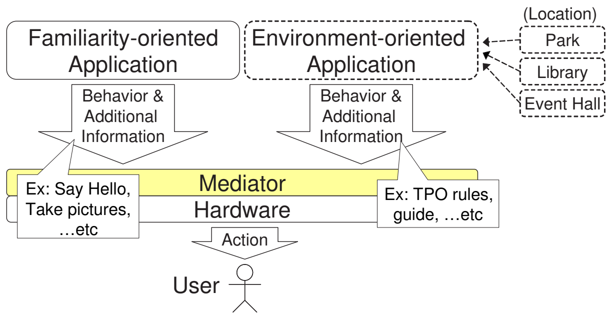
Familiarity-oriented Application
Environment-oriented Application
(Location)
Park
Library
Event Hall
Behavior & Additional Information
Behavior & Additional Information
Ex: Say Hello, Take pictures, ...etc
Mediator
Hardware
Ex: TPO rules, guide, ...etc
Action
User

Figure 2: Mediation between Each Application

As Fig.2 shows, a personal robot uses its original application to represent familiar actions. We call such an application a *familiarity-oriented application*. On the other hand, the robot dynamically loads a new application to perform an information service in the current environment. We call such an application an *environment-oriented application*. Usually applications and the actions of robots are traditionally called a behavior. In this paper, we call a sequence of requests for robot motions from an application a *behavior*. In addition, we name a sequence of robot motions which appear to the user an *action*. A behavior conflicts with another behavior requested from the counterpart. Each application should request the mediator to protect these behaviors from interference. The mediator needs information about what is critical for the behaviors. We call this information *additional information*.

2.2 Features of Applications

The mediator's job is to take care of the difference between both applications in order to solve the interference. Familiarity-oriented applications are usually designed with robot hardware by the same robot designer. These applications provide familiar actions, which the user finds comforting. Therefore, the applications must include many operations to drive motors in order to achieve delicate motions. The mediator should keep the minute features of these motions.

On the other hand, developers of the environment-oriented applications tend to write code with little information about the robots' physical nature. Therefore, predicting the details of robots accompanied by their owners is impossible. The programmers must write their codes to command a robot in an abstract fashion, such as in navigation from one place to another. The environment-oriented applications need to accomplish useful services, and these applications must follow TPO rules more seriously than to just duplicate the details of actions. The mediator should support basic functions for a mobile robot, and these functions must be robust.

2.3 Mediation Requirements

Mediated actions must fulfill specific requirements of the users by estimating the trade-off between the requirements mentioned in Section 2.2. The mediator evaluates the quality of the performance of the robot's behavior from three aspects, i.e., *reliability*, *familiarity*, and *efficiency*. Reliability guarantees keeping the user of the robot and the robot itself secure, in order to accomplish the services provided by an environment-oriented application, and to follow TPO rules in the environment. Reliability is a "must" in a robot's performance. For example, if the behavior generated from another application conflicts with speech recognition, for example, or breaks the context of interaction, the mediator must solve these problems. Reliability also contains the use of reactive actions, since such actions are needed for a robot to keep its user and itself secure.

Familiarity is the degree of owner satisfaction, based on how much the owner is impressed by the robot's actions. Efficiency is measured by the time needed to accomplish the robot's actions. The user hopes his/her robots take familiar actions and accomplish requested tasks in a matter of minutes. However, trade-offs among the aspects exist; e.g., the familiarity of a robot is high if it often performs its familiarity-oriented behavior together with environment-oriented behaviors. In this case, mixing the both behaviors lowers efficiency in terms of the performance of the given task. Mediation provides a way of selection in the trade-offs. In this paper, we adopt the policy that familiarity should be preferred to efficiency as long as familiarity maintains a certain degree of reliability, because a personal robot should be familiar with its user.

3 Mediation Policy

3.1 Familiarity

We suggest a hypothesis on the sources of familiarity so that the mediator can objectively measure *familiarity*, which is the user's impression of the robot. We assume that the robot satisfies familiarity best when the robot performs familiarity-oriented behaviors faithfully to the original action of the familiarity-oriented application. However, the mediator may suspend or divide, or reject some behaviors depending on the various situations facing the robot. From a practical point of view, deciding the guidelines for a robot to perform with better familiarity is reasonable. We assume that the following factors increase the familiarity aspect;

- Similar motions in familiarity-oriented behavior and its total performance time
- Minimizing the suspended time of the familiarity-oriented behaviors
- Seamless switching between both of behaviors
- Responding as quickly as possible to the user

We place importance on the suspended time of familiarity-oriented behaviors. The user's impression of whole behaviors presented in a certain time span must depend on the suspended time. When the suspended time is long, the users' feel unsatisfied. We make three assumptions on the unsatisfactory feelings of the user:

- Dissatisfaction increases at an exponential rate as the suspended time increases.
- Familiarity decreases dissatisfaction.
- Dissatisfaction becomes little, even if the executed part of familiarity-oriented behaviors is little.

Based on the assumptions directly above, our mediation policy is to minimize both the summation and maximal value of the distance.

3.2 Mediation Layers

One of the following four layers of mediations is selected whenever the mediator gets requests from one of the applications while executing requests of the other. A higher layer is more familiar and efficient because the management of time for execution is more intelligent. In general, more programmers' efforts are required to realize more familiar mediation. The mediator tends to select a higher layer, as far as both behaviors and additional informations permit.

Semantic Execution: Both behaviors requested by each application run efficiently at the same time. The mediator understands the semantics of the behaviors and generates new actions to fulfill both of the behaviors.

Concurrent Execution: Both behaviors run simultaneously. The mediator fuses the motions commanded and cuts out a part of the behaviors to exclude the part of behaviors which conflicts with requests from the other applications.

Time-Sharing Execution: Each behavior runs exclusively at one time. The mediator makes a time schedule for exclusive execution. The mediator interrupts running behaviors, or resumes the suspended behaviors on schedule.

Sequential Execution: Each behavior runs using batch processing. The mediator injects interruption, and follows the sequence to duplicate the details of actions.

In the layer of *sequential execution*, a robot becomes familiar to its user, when the robot uses the original motions, which require critical control of motors. If the programmer desires this type of mediation, he/she must describe the start and the end of behavior clearly and should request the layer for short term behaviors. If many devices are used by the robot in the same application, both efficiency and familiarity will decrease substantially.

In the layer of *time-sharing execution*, the user is given an environment-oriented service that contains occasional familiar interactions in addition to sequential ones. This type of mediation allows the robot to behave more familiarly with the user than the sequential execution does because the suspended time of the familiarity-oriented behaviors becomes less, than that of the other. The programmer is required to describe additional information that defines what is critical for the behaviors. The mediator must decide whether or not the current behavior can be suspended. When the robot suspends the current behavior, the mediator must automatically decide when the robot should resume it. When the robot does not suspend the current behavior, the mediator must choose for the robot to reject the new request, or to stores this request on a queue for requests awaiting execution.

In the layer of *concurrent execution*, the robot executes both behaviors in parallel. This type of mediation execution allows the robot to behave more efficiently than in the *time-sharing execution* because the robot can reduce the time to accomplish whole actions. The mediator modifies the behaviors to resolve any conflict between the behaviors and the requests of other applications. Two methods to modify behaviors exist. One method is cutting a part of the behaviors. If each part of the behaviors doesn't relate to the other parts critically, the robot can cut the part which conflicts with the other behaviors. For example, a robot can say hello while guiding the user, although it cannot make a gesture for greeting. The other application is fusing both motions. Some outputs of motors can be expressed as a set of motion vectors. The mediator fuses both vectors of each behavior into a familiar and smooth action. If the programmer uses abstract and long-term expressions, the mediator can simulate fused motion. For example, a robot can dance while guiding the user if the robot doesn't go far away from the user. The movement of the robot's legs can be generalized into a two-dimensional vector.

In the layer of *semantic execution*, the robot interprets the semantics of both behaviors based on a knowledge-based system, and creates actions from the semantics. The mediator doesn't support this layer, because this kind of mediation requires a large amount of knowledge in order to cope with unexpected cases.

4 Implementation

4.1 Selection of Mediation Type

The mediator selects the type of mediation among three possible types; *concurrent execution*, *time-sharing execution*, or *sequential execution.* In general, the mediator selects *concurrent execution* if possible because this execution is more *familiar* and *effective* than the other executions. For example, suppose that the robot walks while singing songs. However, two cases exist such that the mediator should not select the *concurrent execution* in order to keep the *reliability.*

One case is interference between *behaviors* and *additional information.* As discussed in Section 2.1, each application notifies the mediation of additional information to protect behavior. The additional information includes information about what state is critical for the behavior. When a behavior of an application breaks the critical state requested by the additional information of the counterpart application, the mediator selects the time-sharing execution. For example, when a robot is required to keep quiet now, the robot will not sing for a moment.

The other case is interference between both behaviors. Interference contains the case such that both applications access a device simultaneously. Usually, realizing two behaviors with one device is difficult. For example, a robot cannot shake hands when it holds a cup of coffee in one hand. However, when both applications access to a device for movement, the mediator can fuse both motion vectors of each behavior, because the mediator can simplify the movement of robot into the shift of the robot position in the floor plain. Even if both behaviors interfere with each other, the mediator selects concurrent execution, when both behaviors request the movement of robots.

4.2 Concurrent Execution

In the layer of *concurrent execution*, the mediator fuses trajectories of both applications' motion because a robot should execute both applications efficiently, while maintaining *familiarity.* As we explained in Section 3.1, familiarity contains a human-impression about the similarity between mediated motion and the usual motion of familiarity-oriented applications. We assume that the impression is deeply related with the synchronous motions of multiple outputs for familiarity-oriented applications. For example, let us imagine a situation such that a robot dances while guiding its owner. The mediator modifies the movement of the dance along the way to the destination. If the timing of the mediated movement is different from that of the music and hand gestures for the dance, the owner feels that the motion is unusual and unnatural. The mediator should manage concurrent execution based on the timing-conscious fusion of motion vectors.

The mediator modifies motions based on the following strategies to realize timing-conscious fusion.

1. The mediator supports the description about a sequence of motion. The mediator understands the trajectory from the sequence. The trajectory includes the shift of position, the direction, and the velocity of the robot at every unit of time.

2. The mediator supports the description to request the synchronous motions of multiple outputs. The description contains a request to start at the same time, and to end at the same time. The mediator should maintain the timing during the fusing motion.

3. The mediator rotates the whole trajectory of the local movement at the start position of the trajectory in order to adapt the end position of the trajectory to the trajectory of the environment-oriented application. The mediator maintains the internal movement and the timing of the local motion.

4. The mediator modifies the direction of linear movement. This modification may affect the user's impression little if the velocity and the time of linear movement are maintained.

Fig.3 shows an example of a normal dance trajectory of a familiarity-oriented application.

i). The robot starts dancing and playing music for the dance when its owner orders it to dance and play music.

ii). The robot makes a circular movement.

iii). The robot makes liner movements by turning on its edges.

iv). The robot ends the movement when the music ends.

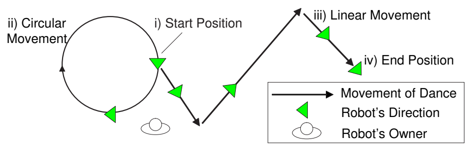

Figure 3: A Normal Trajectory of Dance

Fig. 4 shows the fused trajectory of the dance movement and the navigation movement of an environment-oriented application. In Fig. 4, the owner's destination is on the left side. In Fig. 4, the current position of the robot is on the right side.

I). The robot navigates its owner to his/her destination.

II). When the owner calls his/her robot, the familiarity-oriented application makes the robot stop and turn in the direction of the owner's voice. When the owner orders the robot to dance, the application requests the mediator to dance. The mediator understands that the end of the trajectory is separate from the destination. To make an efficient movement, the mediator makes the robot turn 180-degrees. The mediator starts the dance movement and music to follow the programmer's request to start dancing and playing the music at the same time.

III). The robot makes a circular movement in the same way as shown in Fig. 3.

IV). When the robot makes linear movements, the mediator changes the turning degree of each edge of the trajectory to get closer to the destination.

V). The robot ends the dance at the end of the music.

VI). The robot restarts the navigation.

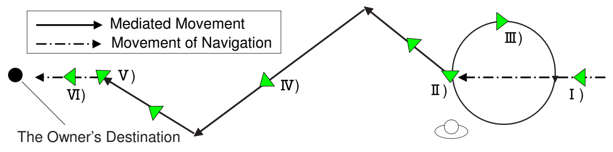

Figure 4: Timing-Conscious Fusion

4.3 Time-Sharing Execution

The mediator supports a programming style, called *embodiment-based exception handling* to express the requests for protecting robots' actions [6]. As we explained in section 3.2, the mediator needs some additional information for the *time-sharing execution.* For that purpose, the mediator should understand (1) where the programmer needs to protect the robot's action in his/her code, (2) what he/she needs to protect, (3) whether he/she needs to refuse the counterpart, and (4) what the programmer needs to do when protection is broken. An exception handling consists of four factors programming constructs, *block, definition, seriousness, and handler.*

A *block* is defined by an area of an application code. The programmer describes it using a curly brace. The robot cannot accomplish certain behaviors because of the actions of the counterpart application. The counterpart application may break the critical control of what, cause user confusion, or interfere with sensing. The programmer should signify the block where the behavior needs a particular state of the robot in its codes.

A *definition* is defined by a state of the robot or by a *refusal exception.* A programmer describes the state of the robot using the reference model. When the robot enters the exception state in the block, the mediator throws the defined exception, and moves the control to the handler associated with the exception. The refusal exception is defined as a special state by the mediator. The refusal exception is thrown when the requested command is refused because of the conflict with the counterpart application.

Seriousness defines whether the application can compromise or request the refusal of the counterpart. If an exception is defined as serious, the mediator assigns that exception as having high priority to protect it from the exception with restricting requests by the counterpart. A *handler* defines what the application should do, and when the exception signals are thrown. One handler is associated with one definition of the exception. The programmer should describe exception handling so the robot can take correct action.

The programmer may require the robot to make more faithful actions than those of the *time-sharing execution.* The mediator also allows the programmer to require the *sequential execution* using these programming constructs. If the programmer defines any access of the other application as the definition of an exception, the mediator selects the *sequential execution.*

Of course the mediation result depends on the programmers' descriptions. However, we can assume that a programmer will not use serious exceptions frequently because too much occupation of robot resources reduces the familiarity of the robot.

5 Simulation

5.1 Assumption about Robots' Abilities

This section describes the simulation results required to prove that this system can improve familiarity. Suppose that the robot in the simulation has the following generic abilities of mobile robots: The robot can move at a rate of 150cm/s, which is faster than the normal walking speed of humans. The error of movement is very low because the generic mobile robot has odometers and moves correctly using the information for moving a short distance.

The generic mobile robot has sensors to localize itself, and the mediator has a general localizing method using the sensors. In this simulation, we assume that the mediator has an indoor navigation method based on ceiling images [9]. This method can be applied to various indoor environments using an inexpensive camera. Assume, therefore, that the robot has a camera to look up at the ceiling of corridors. The environment-oriented application has a map of the ceilings of the building that readies the application. The mediator can localize the robot, because it can match the current view of the camera with the map. The robot navigates at a rate of 100cm/s, which is the normal walking speed of humans, and is under the maximum speed of the robot. We assume that the robot can execute matching at enough frame-rates to move the navigation speed.

The mediator can make the path to the goal with the information of walls described in the map using Voronoi graph methods [2]. The mediator makes a sub-goal at the corner of the path. The robot stops at every sub-goal, and turns at the corner. The robot can also generate a smooth trajectory and does not need to stop except for the sub-goals. As a consequence, we assume that the drive distance is equal to the theoretical distance needed for reaching the goal.

5.2 Original Action of Both Applications

We assume the actions of both applications as in the following. Fig. 5 shows the action of the environment-oriented application if the familiarity-oriented application has no action. The application makes the robot help to navigate the guests from an elevator to a laboratory through the corridor. This navigation takes 39.6 seconds through the desirable trajectory.

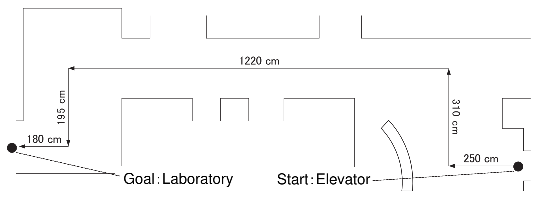

Figure 5: Experimental Environment

Suppose that the familiarity-oriented application makes the robot dance as depicted in Fig. 3. When the owner makes a loud sound, the robot turns to the direction of the sound using a microphone array. The robot says, "May I help you?" If the robot recognizes the keyword "dance", the robot says "OK!" and starts dancing, as in Fig. 3. When the robot ends the dance, the robot says "the dance has ended." This process usually takes 33 seconds. The familiarity-oriented application requests the exception block during the interactions before the dance. The exception is serious, and requests that the robot keep quiet and stay in the current position to communicate with the owner correctly.

5.3 Simulation Results

Fig. 6 shows the simulation results of the mediated movement of the applications as mentioned in Section 5.2. In this case, the mediator selects *concurrent execution* during the dance. In order to clear the effect of each mediation method, we simulate the case of the *time-sharing execution* (Fig. 7), and the case of the *sequential execution* (Fig. 8).

The owner calls the robot in the way of the navigation as shown in Fig. 6. 21.7 seconds pass since the robot starts the navigation. When the robot hears that the owner is calling it, the robot stops the navigation and turns to the owner. The robot communicates with the owner as explained in Section 5.2, which takes 9 seconds. The robot starts dancing as shown in Fig. 4. The dance takes 24 seconds, and the robot moves 346.4cm toward the goal as shown in Fig. 5. The robot reports the end of dance, and restarts the navigation. This navigation takes 18.4 seconds to reach the goal.

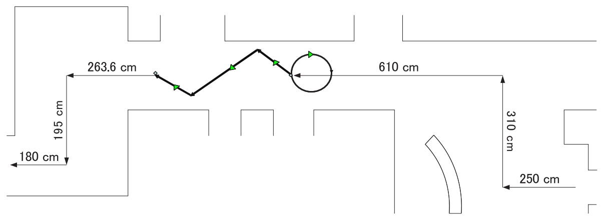

Figure 6: Concurrent Execution

In Fig. 7, the owner calls the robot at the same time as in the example in Fig. 6. In the time-sharing execution, the robot stops the navigation during the dance. The robot plays the dance faithfully to the familiarity-oriented application. The location between the robot and its owner is the same as shown in Fig.3. In consequence of the dance, the robot moves 346.4cm toward the start position.

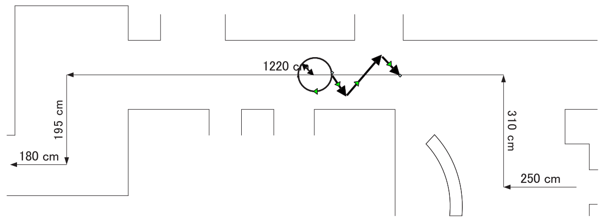

Figure 7: Time-Sharing Execution

If the mediator selects the sequential execution, the robot moves as shown in Fig. 8. In this case, the robot doesn't stop the navigation, even if the robot hears that the owner is calling it. The robot starts the interactions after the navigation. Consequently, the owner waits 17.9 seconds for the robot's reaction.

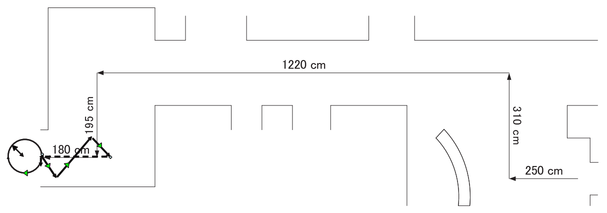

Figure 8: Sequential Execution

In this paper, we only simulate cases where the mediator mediates both actions. The robot executes an action without any consideration of the counterpart. Therefore, the robot may fail to recognize the owner's requests, or the robot may make a movement that doesn't look like part of the dance.

Table.1 summarizes the results of the simulation. Responsibility represents the time necessary for a robot's reaction to the owner's order of the familiarity-oriented application. The more responsibility a robot has, the smaller the suspended time of familiarity-oriented behaviors becomes. As explained in Section 3.1, suspended time is deeply related to familiarity. In the case of the time-sharing execution and the concurrent execution, the robot turns to the user as soon as possible when it hears that the owner is calling it. In consequence, the concurrent execution and the time-sharing execution have an advantage regarding familiarity. Efficiency is the entire time required to accomplish the owner's orders. Concurrent execution is the most efficient execution of the executions.

	Responsibility	Efficiency
Concurrent	0	73.1
Time-Sharing	0	80.4
Sequential	17.9	75.6

Table 1: Simulation Results

6 Discussions

In the simulation in Section 5.3, we should measure the modification of the robot's motion from the usual motion in order to strictly estimate the familiarity-oriented behavior. The robot behaves more efficiently if the mediator transforms the trajectory more than from the original trajectory. However, the transformation makes the robot somewhat unnatural from the owner's point of view. We estimate the appearance of this modification by the following parameters: synchronous motion with other outputs, the shape of the trajectory, and the distance and direction of the robot from the owner. The mediator keeps the applications synchronous because the mediator aims for a timing-concise fusing motion. The mediator also keeps the shape of the trajectory in substance. If the difference in the direction of linear movement is small, then the owner will feel that the action is natural. The distance and direction of the robot from the owner may change a lot from the naturally-perceived action. A long action brings about a great difference of the distance and direction. The mediator should cut a long action into some parts, in order to modify the distance and direction at the cut points.

7 Related Work

Few attempts have been made to give robots the ability to execute parallel multi-applications developed independently.

In this paper we have tried to measure human impressions to decide the policy of mediation. Imai and Kanda et al. [4, 5] have discussed human-robot interactions from the viewpoint of human impression expression. Their research discovers evaluative methods of human-robot interaction (both subjective and objective measurements), analyzes social relationships between humans and robots, in order to introduce robots to actual human society.

The mediator should be applied to various hardware. Recent research of middleware for robots will help us to deal with the various hardware. ORiN [8] standardizes the specifications of interface and data, as well as standardizing negotiation for robot controllers to communicate with various applications. This research allows heterogeneous robots to communicate with each other, and separates applications from robot controllers.

OPEN-R [3] is a re-configurable robot platform for the robot entertainment system. This platform analyzes a robot configuration in syntax and gives semantics to robot components. Based on OPEN-R, a program is portable to robot hardware without changing codes.

8 Conclusion

This paper has described several factors required for extending the scope of personal robots. We established familiarity among the requirements and suggested a policy of mediation for personal robots. We introduced four methods for mediation and explained the mechanism to realize three of these methods. We proposed a timing-conscious fusing motion to realize concurrent execution, and proposed an exception-handler to realize time-sharing execution. We simulated each mediation method, and proved that the proposed system improves familiarity.

References

[1] J. Buhmann, W. Burgard, A. Cremers, T. H. D. Fox, F. Schneider, J. Strikos, and S. Thrun. The Mobile Robot Rhino. *AI Magazin*, 16(1):31–38, 1995.

[2] H. Choset, K. Nagatani, and A. Rizzi. Sensor Based Planning: Using a Honing Strategy and Local Map Method to Implement the Generalized Voronoi Graph. In *SPIE Mobile Robotics*, 1997.

[3] M. Fujita, H. Kitano, and T. Doi. Syntactic-Semantic Analysis of Reconfigurable Robot. In *Proc. of the 1999 International Conference on Intelligent Robots and Systems (IROS)*, pp. 1567–1572. IEEE/RSJ, 1999.

[4] M. Imai, T. Kanda, T. Ono, H. Ishiguro, and K. Mase. Robot Mediated Round Table: Analysis of the Effect of Robot's Gaze. In *Proc of The 11th International Workshop on Robot and Human Communication (RO-MAN)*, pp. 411–416. IEEE, September 2002.

[5] T. Kanda. *A Constructive Approach for Communication Robots*. PhD thesis, Kyoto University, 2003.

[6] A. Kobayashi, I. Kume, A. Ueno, Y. Kono, and M. Kidode. A robot programming model for mediating between familiarity-oriented behaviors and environment-oriented behaviors. In *Proc. the 7th of the World Multi-Conference on Systemics, Cybernetics and Informatics (SCI)*, 2003. to Appear.

[7] T. Matsui, H. Asoh, J. Fry, Y. Motomura, F. Asano, T. Kurita, I. Hara, and N. Otsu. Integrated Natural Spoken Dialoge System of Jijo-2 Mobile Robot for Office Services. In *Proc. of The 16th National Conference on Artificial Intelligence (AAAI-99)*, Florida, July 1999.

[8] M. Mizukawa, H. Matsuka, T. Koyama, T. Inukai, A. Noda, H. Tezuka, Y. Noguchi, and N. Otera. ORiN: Open Robot interface for the Network, The Standard Network Interface for Industrial Robots and its Applications. In *Proc. of International Symposium on Robotics (ISR)*, 2002.

[9] H. Tani, Y. Matsumoto, and T. Ogasawara. Indoor Navigation Based on Ceiling Images - Automatic Mosaic Method for Building Ceiling Maps-. In *Proc. The 19th Annual Conference of the Robotics Society of Japan*, pp. 1013–1014, 2001. in Japanese.

[10] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. NINERVA: A Second-Generation Museum Tour-Guide Robot. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, pp. 1999–2005. IEEE, 1999.