Incorporation of a Software Robot and a Mobile Robot Using a Middle Layer

Tae-Hun Kim, Seung-Hwan Choi, and Jong-Hwan Kim, *Senior Member, IEEE*

***Abstract*—One of the key components of an ubiquitous robot (Ubibot) is the software robot (Sobot), which can communicate with embedded robot (Embot) and mobile robot (Mobot). Sobot is a virtual robot, which has the ability to move to any place or to connect to any device through a network without any spatial limitation. Embot has the capability to sense the surroundings and to interpret the context of the environment and can communicate with Mobot and Sobot. Mobot provides an integrated mobile services. To incorporate Sobot, Embot, and Mobot reliably as an Ubibot, a middle layer is needed to arbitrate different protocols among them. This paper focuses on incorporating Sobot and Mobot to overcome physical limitations of Sobot for physical behaviors in real situations. To implement the incorporation of them, the basic concept and structure of the middle layer are proposed. The effectiveness of the middle layer for Sobot and Mobot is demonstrated through real experiments.**

***Index Terms*—Middle layer, mobile robot, software robot, ubiquitous robot.**

I. Introduction

THE UBIQUITOUS robot (Ubibot) is composed of a software robot (Sobot), an embedded robot (Embot), and a mobile robot (Mobot). Sobot is a virtual robot, which has the ability to move to any place through the networked devices. Embot is implanted in the environment or upon Mobots. It gathers information from various sensors. Mobot is a mobile robot that can make behaviors in the real world using real actuators [1], [2].

Ubibot represents the cutting edge of technology with the advent of the ubiquitous era. To implement the concept of "ubiquitous," the importance of Sobot is increasing. It can be in any desktop computer, Personal digital assistance (PDA), or mobile phone. So human beings can interact with Sobot very easily. Since Sobot is a software-based robot, it is easy to change its graphical appearance according to user's preference. Its appearance can, thus, be made human friendly. Sobot can, therefore, be used for entertainment, education, psychological treatments, and so on.

In human–robot interaction (HRI), there are four paradigms, as described in [3]. A robot can be viewed as a tool, a cyborg extension, a sociable partner, and an avatar. It can be used as a tool to perform a simple task. It can be regarded as a cyborg extension, which can be a part of human body for disabled people or for enhancing physical ability of human beings. It can also be a partner, which has sociable abilities such as emotions, intelligences, and so on. In this paradigm, a robot must be able to interact with people by understanding what people want and by providing social cues for them to understand what its intention is. Lastly, human beings can communicate with people who are in a remote site, using a remote robot as an avatar or a software robot representing them as if they are in a remote site. A Sobot-based "detect and alert" system called *NewsAlert* [4] delivers Internet alerts to the desktops of managers and executives in the form of a personalized electronic newspaper. Verbots [5] are also avatars with a combination of artificial intelligence, natural language processing, and creativity. They allow users to create an engaging virtual personality, and they can talk to users by changing the agent's voice, pitch, and speed. Sydney [6] is a virtual pet and exists in virtual world. It can get real information through the vision system and the voice system called DogEar. But they are restricted to sensing. Even though it can see real objects and can hear the human voice commands, it only displays its behaviors on the computer screen.

Since software-based robots are in virtual environments, they have physical limitations to serve human beings in real situations, to interact physically and naturally with them, or to make their own physical behaviors. To overcome these physical limitations, Sobot must be able to use Mobot that has a physical body, actuators, and sensors. However, as there will be various Mobots and Sobots in ubiquitous environments, it is impossible to consider all the specific architectures of Mobots and Sobots. This leads to the necessity of a standard protocol between Mobots and Sobots.

For this purpose, this paper proposes a middle layer to incorporate Sobots and Mobots. The middle layer is an interface between them. Key components of the middle layer are the sensor mapper and behavior mapper. The sensor mapper helps Sobot get physical sensor information from Mobot. Thus, Sobot in a virtual environment can use physical information. Behavior mapper helps Sobot make physical behavior using Mobot in a real environment. Thus, Sobot can show physical behaviors and interact physically with human beings, and so on in real environments. This paper also presents a control arbiter. Since a user, Sobot, and Mobot itself can control a single body (Mobot) at the same time, arbitration of control commands are needed. Thus, the control arbiter is used for arbitration of control commands from Sobot, Mobot, and a user.

This paper is organized as follows. Section II presents the middle layer. In Sections III and IV, the sensor mapper and behavior mapper are proposed in detail, respectively. Section V describes control arbiter for coordinating commands from Mobot, Soobt,

Manuscript received April 13, 2006; revised September 20, 2006. This work was supported by the Ministry of Information and Communications, Korea, under the Information Technology Research Center Support Program. This paper was recommended by Associate Editor M. Funabashu.

The authors are with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Korea (e-mail: thkim@rit.kaist.ac.kr; shchoi@rit.kaist.ac.kr; johkim@rit.kaist.ac.kr).

Digital Object Identifier 10.1109/TSMCC.2007.905850

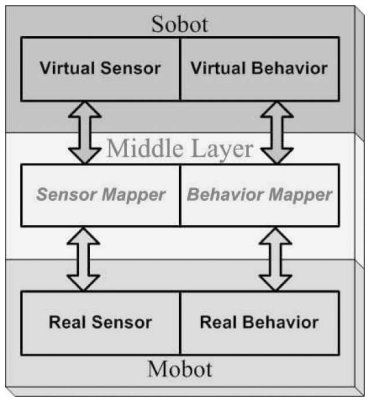


Fig. 1. Middle layer is for interface between Sobot in virtual environment and Mobot in real environment.

Fig. 2. Data structure of sensor information.

and the user. In Section VI, experimental results are described. By employing the middle layer, Sobot can get physical information from Mobot and make its emotional expressions using Mobot in the real world. Also, the importance of the control arbiter is demonstrated. Finally, concluding remarks follow in Section VII.

II. Middle Layer

Sobot has virtual sensors and generates its behaviors in a virtual environment. With virtual sensors, it can acquire virtual information, i.e., time, distance, light intensity, temperature, and so on. Based on perception using those sensors, it performs behaviors such as moving forward, turning left, etc. using a virtual motor system in a virtual environment. In contrast with Sobot, Mobot obtains sensor information from real sensors attached to its body and executes behaviors through a real motor system. The middle layer is for the interface between Sobot and Mobot, as shown in Fig. 1. It is needed to incorporate Sobot and Mobot, which enables Sobot to provide physical services to human beings through Mobot.

The middle layer consists of the sensor mapper and behavior mapper. The sensor mapper helps Sobot to get real information from real sensors that are attached to Mobot in the real environment. The behavior mapper also enables Sobot to make physical behaviors using real actuators of Mobot in a real environment.

III. Sensor Mapper

The role of the sensor mapper is to map physical sensor information from Mobot to virtual sensor information of Sobot when Sobot moves to Mobot. There are three procedures.

A. Sensor Registration

When Sobot moves to Mobot, Sobot registers its virtual sensor information onto the sensor mapper, and also Mobot registers its real sensor information onto the sensor mapper. The data structure of sensor information is shown in Fig. 2. Each item is described as follows.

1) *ID:* As Sobot and Mobot have various sensors, their sensors should be identified so that the sensor mapper can use them. Even though the same kind of sensors is attached to a Mobot, they should also be identified. Thus, ID numbers are assigned to all the sensors.
2) *Type:* This is for defining basic function of sensors. This paper uses "DISTANCE" and "VISION" sensor types. The sensor whose type is "DISTANCE" measures distance between a robot and an obstacle. The sensor whose type is "VISION" makes facial recognition. A robot can recognize its master with the vision sensor. The sensor type can include various sensors such as "TEMPERATURE," "PRESSURE," etc.
3) *Location:* It has the location information of each sensor of Mobot and Sobot. Some information from sensors such as temperature sensor and humidity sensor are not seriously dependent on the location where it is mounted because the variation of temperature within the robot's vicinity is negligible. However, information from sensors such as ultrasonic sensor or infrared sensor is seriously dependent on the sensor location. For example, if the location of ultrasonic sensor is in front of a robot and the measured value from that sensor is 10 m, it means obstacle is 10 m away in its front. But if the sensor is in its rear and the same data are obtained, it means obstacle is 10 m away in its rear. So the location of sensor is very important.
4) *Value:* This is the physically measured value from sensors of which units are in MKS. Exceptionally, the vision sensor value is 1 when the master face is recognized, otherwise 0.

B. Sensor Mapping

Once a sensor mapper obtains the sensor information of Sobot and Mobot, it starts to map physical sensor information onto virtual sensor information. The sensor mapper has two comparators, type comparator and location comparator, as shown in Fig. 3. The type comparator is to match the sensor type between virtual and physical sensors. The inputs of type comparator are one of virtual sensor types and one of physical sensor types to be compared. If both sensor types are not matched, it terminates the comparison and starts to compare another pair of sensor types. But if both sensor types are matched, the output of type comparator is "true," and this is used for validation of values in mapping table (Table I).

And then, the location comparator is conducted to match the physical sensor location information with the virtual one. If both sensor locations are matched, the output of the location comparator is "true," and if not, "false" is the output. In this way, the sensor mapper matches the physical sensor information with the virtual one in sequence. Once the sensor mapper completes all the possible comparisons, a mapping table is created. Table I shows the mapping table comprising three parts.

1) *Mapped virtual sensor ID:* This is a mapped virtual sensor ID related to a physical sensor ID. The entry may be also

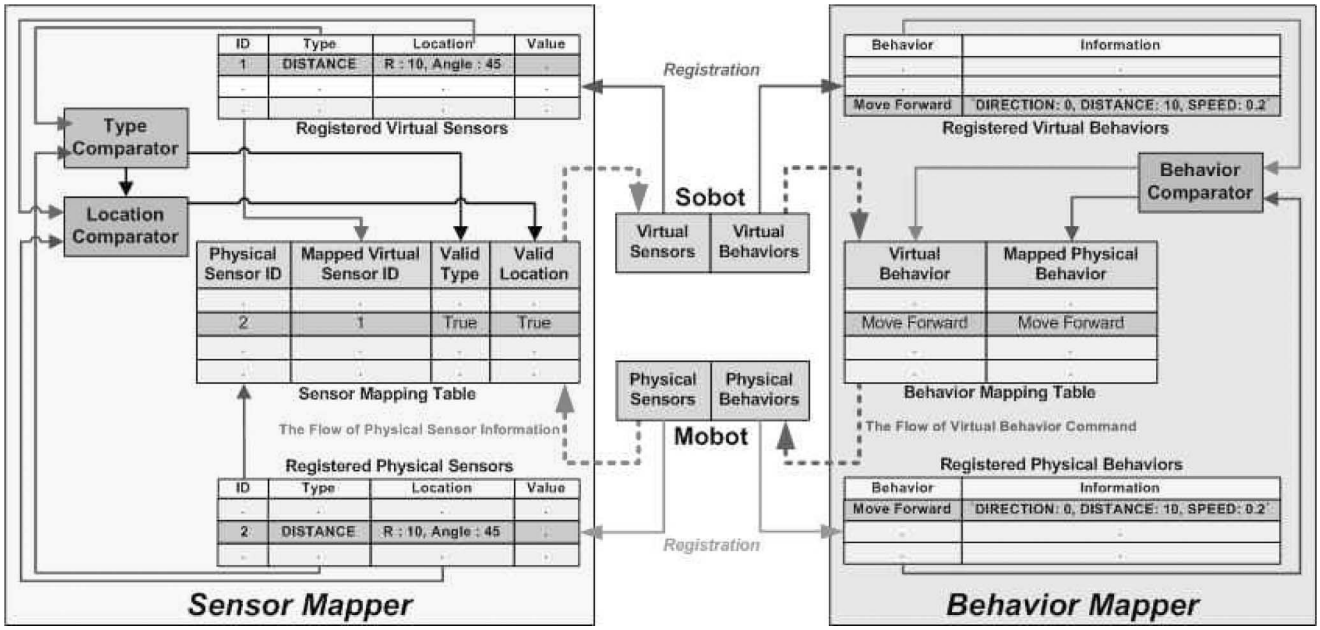


Fig. 3. Sensor mapper and behavior mapper.

TABLE I
SENSOR MAPPING TABLE

Physical sensor ID	Mapped virtual sensor ID	Valid type	Valid location
1	3	True	True
2	1	True	False
3	None	False	False

"none" since there is no virtual sensor whose function is the same as that of the physical sensor. In this case, Sobot cannot get physical information from that physical sensor.

2) *Valid type:* This value is "true" if the output of the type comparator is "true." If this value is "true," Sobot can get physical information from that sensor.
3) *Valid location:* This value is "true" if the output of the location comparator is "true." This means that the type and location of physical sensor are the same as those of the virtual sensor, respectively. Then Sobot can get physical sensor information.

C. Sensor Data Transfer

Once the mapping table is created from the sensor mapper, it is not updated anymore. In other words, it means the sensor mapper does not make any comparison, and instead, just looks up the mapping table when it needs to transfer physical sensor information to Sobot. For example, if a mapped virtual sensor exists and both valid type and valid location are "true" in the sensor mapping table, the measured value and location of the physical sensor is transferred to Sobot as those of the mapped virtual sensor. This happens as soon as one of physical sensors makes a new measurement. In this way, Sobot can get real sensor information in the environment.

However, if a mapped virtual sensor exists, valid type is "true," but valid location is "false," only the measured value of the physical sensor is transferred to Sobot. Its location information is not transferred. Instead, "null" is transferred. Wrong location information may mislead Sobot such that it may cause a serious problem.

IV. BEHAVIOR MAPPER

Sobot can dance and move in a virtual environment. Only graphical behaviors can be seen on the computer monitor. It means that Sobot has physical limitations to provide physical service and to make its own physical behaviors. To overcome this limitation, Sobot has to use Mobot that has a physical motor system. The role of the behavior mapper is to map Sobot's behavior to physical behavior of Mobot. However, Mobot's behaviors cannot be exactly the same as Sobot's, as they are dependent on its hardware and mechanical configurations. Also, it may not be possible for Sobot to control every actuator in Mobot. To solve this problem, Mobot should have abstract modular behaviors. That is, each modular behavior should be independently provided by controlling related actuators using its own control methods, as shown in Fig. 4. Thus, Sobot does not have to know about every actuator attached to Mobot. It needs the behavior interface module with Mobot.

A. Behavior Registration

The behavior mapper must have information on virtual and physical behaviors from Sobot and Mobot to map Sobot's behaviors to physical ones. When Sobot is to move to Mobot, both robots should register their behavior information to the behavior mapper. Thus, the data structure of behavior information is needed for registration, which consists of *behavior* and *information*, as shown in Fig. 5(a).

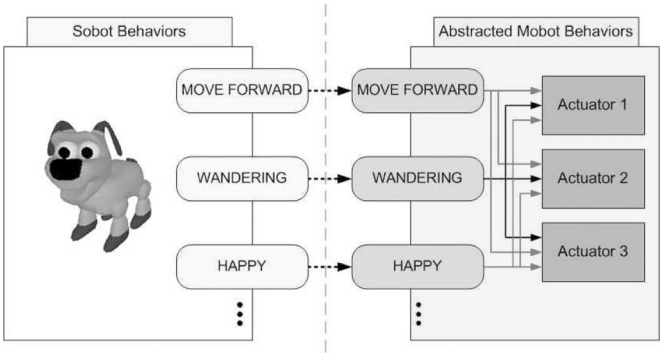


Fig. 4. Sobot behaviors and abstracted Mobot behaviors.

Behavior	Information

(a)

'MOVE FORWARD'	'DIRECTION: 0, DISTANCE: 10, SPEED: 0.2'

(b)

'EMOTION'	'Happy'

(c)

Fig. 5. Data structure of behavior information.

TABLE II
BEHAVIOR MAPPING TABLE

Virtual Behavior	Mapped Physical Behavior
EMOTION (HAPPY)	EMOTION (HAPPY)
WANDERING	WANDERING
MOVE FORWARD	MOVE FORWARD

1) *Behavior:* This is the name of behavior module, represented by a string such as "MOVE FORWARD," "MOVE BACKWARD," "EMOTION," and so on.
2) *Information:* This contains the information that the behavior module uses. For example, in Fig. 5(b), "MOVE FORWARD" behavior has information about the direction which Mobot has to face, the distance which Mobot has to move to the given direction, and the speed which Mobot has to keep while moving. In Fig. 5(c), "EMOTION" behavior has information about which emotion should be expressed.

B. Behavior Mapping

The behavior mapper uses a registered data structure of behavior information to map Sobot's behaviors to physical ones. One of the key components of the behavior mapper is the behavior comparator, as shown in Fig. 3. It takes two inputs. One is *virtual behavior* from Sobot, and the other is *physical behavior* from Mobot. It compares two behaviors from Sobot and Mobot. If two behaviors are the same, the behavior mapper maps the *virtual behavior* onto the *physical behavior* in the behavior mapping table. Once the behavior comparator completes all the possible comparisons, a behavior mapping table is created. Table II shows some of them.

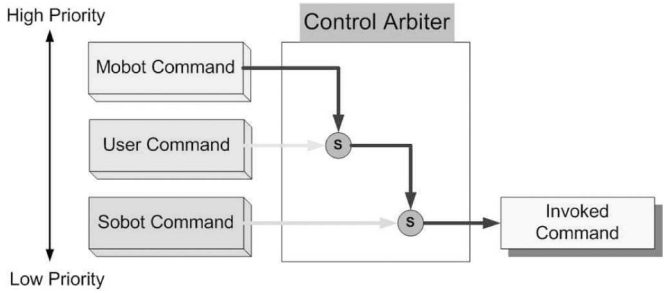


Fig. 6. Control arbiter arbitrates commands from Sobot, Mobot, and user.

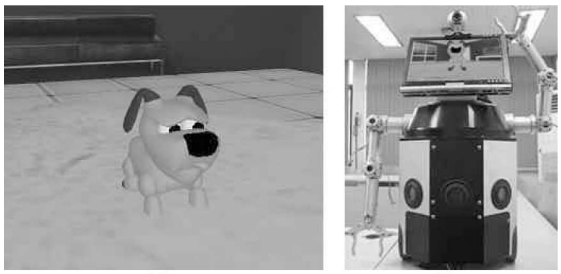
C. Behavior Information Transfer

Whenever Sobot wants to make its behavior in a real environment, it sends a command to the behavior mapper. Then, the behavior mapper looks up the behavior mapping table. If there is a mapped physical behavior corresponding to Sobot's behavior, the behavior mapper transfers *information* to the physical behavior module of Mobot. Then, Mobot makes a physical behavior using the behavior module. The behavior module has its own kinematics and control methods for that behavior. In this way, Sobot can realize its behaviors as physical ones through Mobot in a real environment.

V. CONTROL ARBITER

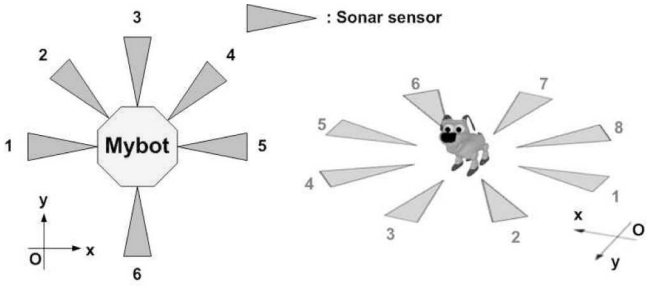
Since Mobot can be controlled by the Mobot itself, Sobot, or user the for their purpose, there may be control collision among them if they are not properly coordinated. Thus, arbitration of control is needed to avoid such collision. As Fig. 6 shows, the control arbiter arbitrates controls from Sobot, Mobot, and the user with a priority-based arbitration similar to the subsumption architecture [10]. If control commands from Sobot, Mobot, and the user are invoked at the same time, the control command from the one with the highest priority suppresses control commands from the others with lower priorities. In this way, only one control command is delivered to the behavior module at a time.

In this paper, the priority of Mobot control is assumed to be the highest because the owner of the hardware system is Mobot, and it has to protect itself from dangerous situations such as collision, falling down, and so on. To enable this kind of protection, the reactive system is adopted to control Mobot as the response time is very short [7]. The priority of user control is the second highest. The user can control Mobot remotely for their own purpose. For example, the user might want to control Mobot to get remote images from a universal serial bus (USB) camera attached to Mobot. The priority of Sobot control is the lowest because Sobot manages Mobot in a normal state mainly for services for the user. If the priority of user control is lower than that of Sobot control, the user may not be satisfied with the delayed response caused by Sobot. Based on this priority, the control arbiter can arbitrate controls from Sobot, Mobot, and the user without any control collision.



(a) (b)

Fig. 7. (a) Sobot (called Rity) and (b) Mobot (called Mybot) were used in experiments.



(a) (b)

Fig. 8. Sensors and their IDs are distributed around the body.

VI. Experiment

These kinds of experiments were carried out to demonstrate the effectiveness of the proposed scheme. One was about the sensor mapper, another about the behavior mapper, and the other about the control arbiter. A Sobot called Rity [Fig. 7(a)] and a Mobot called Mybot [Fig. 7(b)] were used for the experiments. Rity is a dog-type software robot in a virtual environment. It has virtual sensors such as vision, sonar, touch, light, touch, temperature, sound, gyro, and time. Using these sensors, it can recognize 47 types of perceptions that influence internal state. The internal state module is composed of motivation, homeostasis, and emotion. Rity chooses a proper behavior based on its internal state. It can exhibit five facial expressions and 77 behaviors [8], [9]. It can move to networked devices without any space limitation [1].

Mybot is a wheeled-type mobile robot with six ultrasonic sensors, one vision sensor, and two arms to express emotion. It has five emotional behaviors to express emotional state such as happiness, sadness, anger and fear, and neutral. Mybot can make facial detection and facial recognition from a vision sensor (USB camera). In this experiment, the middle layer was implemented in Mybot instead of using an independent server system.

A. Sensor Mapper

In this experiment, Rity could get physical information from real sensors attached to Mybot. When Rity moved to Mybot, both Rity and Mybot registered their sensor information to the sensor mapper, and then, it started to create the mapping table. The distributions of physical sensor ID and virtual sensor ID are shown in Fig. 8. In Fig. 8(a), sensors 1–6 are ultrasonic sensors, and in Fig. 8(b), sensors 1–8 are ultrasonic sensors. Both Mybot and Rity have a vision sensor.

Tables III and IV show data structures of sensor information of Mybot and Rity, respectively, which were used for sensor mapping between them. In this experiment, the type comparator was conducted for two kinds of sensor type (DISTANCE and VISION) to match the sensor type between virtual and physical sensors, where * denotes arbitrary measured values. Then, the location comparator was carried out to map the physical sensor location information to the virtual one, where the radius information in sensor location on Rity was not considered for the DISTANCE sensor type and • denotes this meaningless value. In Table IV, angle information is provided in the Location entry. Thus, sensor mapper used only angles for the location comparison.

TABLE III
Data Structure of Sensor Information of Mybot

ID	Type	Location	Value
1	DISTANCE	Radius:0.15, Angle : 180	*
2	DISTANCE	Radius:0.15, Angle : 135	*
3	DISTANCE	Radius:0.15, Angle : 90	*
4	DISTANCE	Radius:0.15, Angle : 45	*
5	DISTANCE	Radius:0.15, Angle : 0	*
6	DISTANCE	Radius:0.15, Angle : 270	*
7	VISION	•	*

TABLE IV
Data Structure of Sensor Information of Rity

ID	Type	Location	Value
1	DISTANCE	Radius: • , Angle : 180	*
2	DISTANCE	Radius: • , Angle : 135	*
3	DISTANCE	Radius: • , Angle : 90	*
4	DISTANCE	Radius: • , Angle : 45	*
5	DISTANCE	Radius: • , Angle : 0	*
6	DISTANCE	Radius: • , Angle : 315	*
7	DISTANCE	Radius: • , Angle : 270	*
8	DISTANCE	Radius: • , Angle : 225	*
9	VISION	•	*

TABLE V
Sensor Mapping Table in Experiment

Physical Sensor ID	Mapped virtual Sensor ID	Valid Type	Valid Location
1	1	True	True
2	2	True	True
3	3	True	True
4	4	True	True
5	5	True	True
6	7	True	True
7	9	True	True

For the VISION sensor type, it was assumed that the location of the vision sensor did not matter and the detected objects were just in front of the robots. Thus, in this experiment, the result of the comparison of VISION sensor type was "true." Table V shows the mapping table created by the sensor mapper.

In Fig. 9, green rectangles are the locations of physical ultrasonic sensors and the red circles are virtual obstacles. Rity was getting physical information through real ultrasonic sensors.

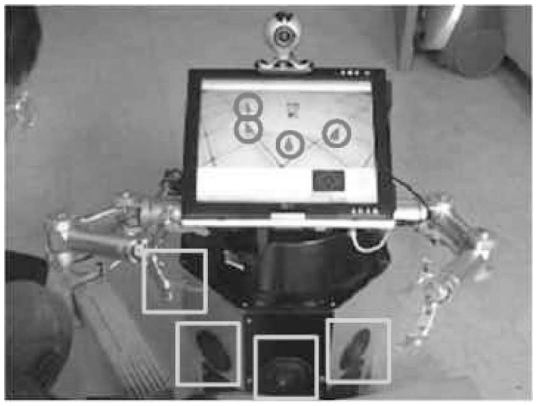


Fig. 9. Rity was getting physical information through real sensors of Mybot.

Fig. 10. Master's face was displayed after facial recognition.

Virtual obstacles were displayed in the virtual environment, which corresponded to physical obstacles detected by real sensors.

In Fig. 10, the red circle is master's face. It means Rity knows that there is a master in its front.

B. Behavior Mapper

This experiment was to show how Rity could make emotional expressions in the real world using Mybot. Rity could express five emotional states such as happiness, sadness, neutral, fear, and anger. Mybot also had five emotional behaviors. The behavior mapper created the behavior mapping table when Rity moved to Mybot like the sensor mapper. It mapped each of the virtual emotional expressions to a corresponding emotional behavior of Mobot.

Table VI shows data structures used for behavior mapping. Table VII shows the behavior mapping table created by the behavior mapper. For example, when Rity wants to express sadness, it sends a command to the behavior mapper. Then, the behavior mapper looks up the behavior mapper table. As there is a mapped physical behavior for sadness in the behavior mapping table, the behavior mapper provides Mobot with the sad behavior module. Fig. 11 shows experimental results for emotional behaviors to express neutral, happy, and sad states of Rity. Mybot was expressing Rity's emotion using its two arms

TABLE VI
DATA STRUCTURE FOR BEHAVIOR MAPPER OF RITY AND MYBOT

Behavior	Information
EMOTION	Neutral
EMOTION	Happiness
EMOTION	Sadness
EMOTION	Fear
EMOTION	Anger

TABLE VII
BEHAVIOR MAPPING TABLE

Virtual Behavior	Mapped Physical Behavior
EMOTION (Sad)	EMOTION (Sad)

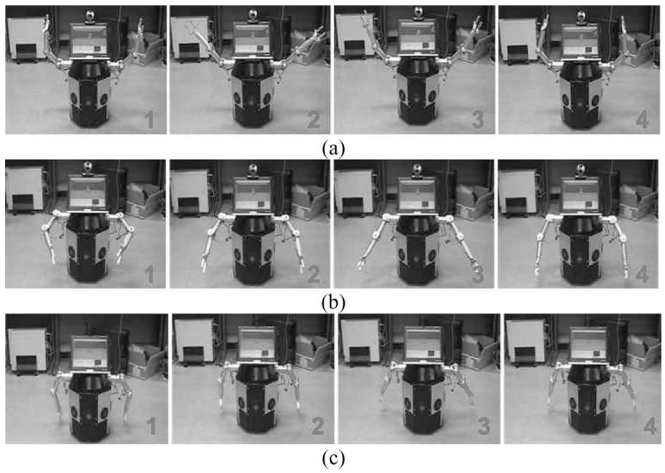


Fig. 11. Rity showed emotional behaviors in the real world using Mybot.

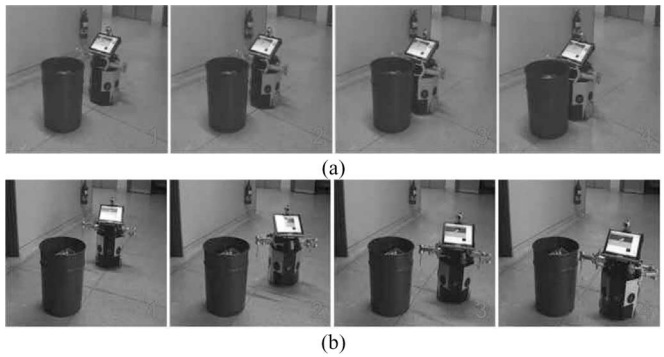


Fig. 12. Demonstrations with and without control arbiter.

in the real world. It means that Rity could express its emotional behavior the real in world using Mybot.

C. Control Arbiter

In this experiment, the role of the control arbiter was demonstrated. The key role is to arbitrate control commands from Rity, Mybot, and a user. Fig. 12(a) shows a situation when the control arbiter was not activated and a user controlled Mybot to move forward without noticing an obstacle in front of it. A few seconds later, Mybot activated the collision avoidance behavior module to avoid the obstacle. Then there became two control commands to make Mybot move. Control commands collided

with each other such that it was not certain which one was delivered to the motor system. In this case, Mybot continued to move forward, and it collided with the obstacle.

In Fig. 12(b), the situation was the same as the previous one except that the control arbiter was activated. If the distance between Mybot and the obstacle was less than the limit range for safety, Mybot invoked a collision avoidance command to avoid collision. At this instant, two control commands became active with their own objectives, but they were arbitrated by the control arbiter based on their priorities. Since the priority of Mybot control was higher than that of user control, the Mybot control command suppressed the user control command. Thus, invoked control was Mybot control, and it was delivered to the motor system. As a result, Mybot could avoid the obstacle.

VII. Conclusion

This paper proposed a middle layer for incorporating Sobot and Mobot. The middle layer was composed of the sensor mapper and behavior mapper. It could help Sobot get physical information from real sensors of Mobot, which led Sobot realize its behaviors as physical ones in the real world through Mobot. The effectiveness of this scheme was demonstrated by carrying out experiments on sensor mapper, behavior mapper, and control arbiter.

In the coming ubiquitous era, Ubibots will be around human beings, and most of interactions will be made through Sobot because of its high accessibility. Sobot requires the functionality to provide physical services to human beings using Mobot. For this purpose, the proposed middle layer will play an important role as an interface between Sobot in a virtual environment and Mobot in a real environment.

References

[1] J.-H. Kim, K.-H. Lee, Y.-D. Kim, N.-S. Kuppuswamy, and J. Jo, "Ubiquitous robot: A new paradigm for integrated services," in *Proc. IEEE Int. Conf. Robot. Autom.*, Rome, Italy, Apr. 2007, pp. 2853–2858.

[2] J.-H. Kim, Y.-D. Kim, and K.-H. Lee "The third generation of robotics: Ubiquitous robot," (Keynote Speech Paper) presented at the Int. Conf. Auton. Robots Agents, Palmerston North, New Zealand, Dec. 2004.

[3] C. Breazeal, "Social interactions in HRI: The robot view," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 34, no. 2, pp. 181–186, May 2004.

[4] D. King and K. Jones, "Competitive intelligence, software robots and the Internet: The NewsAlert prototype," in *Proc. 28th Annu. Hawaii Int. Conf. Syst. Sci.*, 1995, vol. 4, pp. 624–631.

[5] Verbot. (2000). [Online]. Avaiable: http://www.verbots.com.

[6] S.-Y. Yoon, R. C. Burke, B. M. Blumberg, and G. E. Schneider, "Interactive training for synthetic characters," in *Proc. AAAI*, 2000, pp. 249–254.

[7] R. C. Arkin, *Behavior-Based Robotics*. Cambridge, MA: MIT Press, 1998, pp. 205–234.

[8] Y.-D. Kim and J.-H. Kim, "Implementation of artificial creature based on interactive learning," in *Proc. FIRA Robot World Congr.*, 2002, pp. 369–374.

[9] Y.-D. Kim, Y.-J. Kim, and J.-H. Kim, "Behavior selection and learning for synthetic character," in *Proc. IEEE Congr. Evol. Comput.*, 2004, pp. 898–903.

[10] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Trans. Robot. Autom.*, vol. 2, no. 1, pp. 14–23, Mar. 1986.

Tae-Hun Kim received the B.S. degree from Kyungpook National University, Daegu, Korea, in 2004, and the M.S. degree from the Korea Advanced Institute of Science and Technology, Daejeon, in 2006.

He is currently with Hyundai Autonet, Kyoungkido, Korea, where he works for autoelectronics and electronics parts. His current research interests include software robots, mobile robots, and mobile telematics systems.

Seung-Hwan Choi received the B.S. degree in 2005 from the Korea Advanced Institute of Science and Technology, Daejeon, where he is currently working toward the Ph.D. degree.

His current research interests include ubiquitous robotics and evolvable artificial creatures.

Jong-Hwan Kim (S'85–M'88–SM'03) received the B.S., M.S., and Ph.D. degrees in electronics engineering from Seoul National University, Seoul, Korea, in 1981, 1983, and 1987, respectively.

Since 1988, he has been with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, where he is currently a Professor. His current research interests include ubiquitous and genetic robotics.

Prof. Kim is currently an Associate Editor of the IEEE Transactions on Evolutionary Computation and the IEEE Computational Intelligence Magazine. He was one of the cofounders of the International Conference on Simulated Evolution and Learning (SEAL). He was the General Chair for the IEEE Congress on Evolutionary Computation, Seoul, in 2001. His name was included in the *Barons 500 Leaders for the New Century* in 2000 as the Father of Robot Football. He is the Founder of the Federation of International Robot-Soccer Association (FIRA) and the International Robot Olympiad Committee (IROC). He is currently the President of FIRA and IROC.