DOI:10.5392/IJoC.2011.7.1.029

A Development of Gesture Interfaces using Spatial Context Information

Doo Young Kwon
Department of New Media, KGIT
Digital Media City, Seoul, South Korea

Ki Tae Bae
Department of New Media, KGIT
Digital Media City, Seoul, South Korea

ABSTRACT

Gestures have been employed for human computer interaction to build more natural interface in new computational environments. In this paper, we describe our approach to develop a gesture interface using spatial context information. The proposed gesture interface recognizes a system action (e.g. commands) by integrating gesture information with spatial context information within a probabilistic framework. Two ontologies of spatial contexts are introduced based on the spatial information of gestures: gesture volume and gesture target. Prototype applications are developed using a smart environment scenario that a user can interact with digital information embedded to physical objects using gestures.

***Keywords**: Spatial Context, Gesture Interface, Gesture Recognition*

1. INTRODUCTION

With advances in sensor and network technologies, physical environments are getting embedded with computer systems [4]. In recent years many researchers have recognized the value of such new computing environments and have suggested computing paradigms such as mobile computing, ubiquitous computing, and wearable computing. They focus on understanding human capability and maximizing the opportunity to access digital information regardless of time and location.

Various applications have driven the research of a large scientific community including computer vision, human-computer interaction, and computer graphics. For instance, gesture is used as a means of pointer and manipulator in different displays namely desktop monitors, tabletop displays, large wall displays, and immersive VR displays. Gesture is also applied to command execution aiming for complete eyes-free interaction in mobile and wearable computing domains [1]. Although a lot of solid research has been done, it has been mainly designed to test the system performance for gesture recognition. Little is applied to the actual use. Therefore, the success of current gesture interfaces has been restricted to relatively controlled environments requiring a special hardware setup and well-defined gestures. Their practical utility has mostly been ad-hoc and not presented within a generative design framework.

Another challenge of gesture recognition is that in intelligent environments with a large number of devices a large number of interactions are possible as well. There are, however, only a limited number of simple, useful gestures. This is because most users prefer gestures that can easily be performed and remembered. Additional context information can be used to resolve ambiguous recognition, since some gestures are more likely within a specific context. E.g. it is more likely the user wishes to turn on the lights if it is dark outside. Furthermore, gestures can be interpreted differently if they are performed in a different context. Thus, gestures can be reused, which reduces the total number of gestures to be recognized.

This paper presents a spatial context-aware gesture interface that improves the usability of gesture-based inputs by combining gesture information with additional context information. There is a large amount of research dedicated to studying the use of context information in human computer interaction. Obviously, it is a far-reaching and difficult goal to define the general application of the context information to the design of gesture interfaces. Here, the paper focuses on specific context information which is related to spatial objects of gestures (locations and objects of a gesture) easily featured together with 3D spatial gestures. We call this particular context information spatial context. Using spatial contexts, we aim to reduce the recognition error and improve the usability of gestures. In this system, the user performs a 3D gesture, which is typically performed in 3D space. 3D spatial gesture preserves a certain spatial relationship between gesture and the

* *Corresponding author. E-mail : ktbae@kgit.ac.kr*
Manuscript received Mar.04, 2011 ; accepted Mar.21, 2011

environment where the gesture occurs [10a].

This paper is structured as follows. In the following chapter some related work about gesture recognition techniques and context aware systems is resumed. After a short overview of the architecture of context model, we present the spatial context model which supports registration and selection of spatial contexts. Then it explains how a gestural command is recognized using the selected spatial contexts. Dynamic Bayesian Network (DBN) is developed to integrate the result of the gesture recognizer with other context information for command recognition. We developed two prototype applications which use 3D spatial gestures to control functions embedded with physical objects.

The text must be in English. The submitted typeset scripts of each contribution must be in their final form and of good appearance because they will be printed directly without any editing. It is essential that the "camera-ready copies" be absolutely clean and unfolded. The copy should be evenly printed on a high quality (300 dots/inch or higher) laser printer. There should not be corrections made on the printed pages. Your paper must be printed actual size (exactly how it is to appear in the Journals) in two columns. The document you are reading is printed in the format that should be used in your paper.

2. RELATED WORK

Deictic gestures are particularly important for spatial inputs requiring information about the position and spatial location of the user when the gesture is performed. Spatial inputs have been used in different applications. Bolt presented the Put that there [2] where users can interact with an object on a large screen display through pointing. Similarly, virtual reality systems often enable users to point and select the virtual objects using gestures. Recently, spatial inputs have been widely applied to applications for smart environments. With the advance of sensor and network technologies, our environments are designed and programmed to understand gestures automatically to a certain degree. These smart environments understand the user's intention, and provide a set of programmed functions such as turning on/off lights and controlling temperature.

In this smart home scenario, spatial information plays an important role in understanding the gesture correctly. Wilson [11] presented a gesture interface where users can interact with the smart environment using gestures (selecting and controlling appliances through pointing, turning it on/off and moving the volume up/down).

Spatial inputs are also employed to facilitate the use of context in HCI. "Context is the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user" [5]. Dey et al. [6] presented CAPpella (context-aware prototyping environment) designed with the concept of programming by demonstration. In this system, end-users can program desired context-aware behaviors (situation and associated action) by demonstrating them to the system and by annotating the relevant portions of the demonstration.

Jih el al. [9] developed a context-aware elderly care system for smart environments. The system interacts with the elder through a wide variety of appliances for data gathering and information presentation. The system tracks the location and specific activities of the elder through sensors, such as pressure-sensitive floors, cameras, bio-sensors, and smart furniture. The status of the elder is monitored and used to provide appropriate system actions. For example, when sensing that the elder has fallen asleep, the system switches the telephone into voice mail mode, and informs and plays back any incoming messages when the elder awakens.

Chen and Kotz [5] describe two ways of using context: active and passive context awareness. In an active context aware system the application adapts automatically to the discovered context whereas in a passive context aware system the application only presents the context information to the user or saves it for later retrieval. In this paper, we build an active context aware system. We introduce the novel concept of gesture target and gesture volume focusing on contextual information such as spatial user context, temporal context and context history. The system supports context sensitive command recognition, where a command is a combination of a gesture and related context, such as an object the user is pointing at. The context includes the position of the user, the pointed object, daytime and previous commands, but more context information could be added easily due to the extensible system design of a DBN.

3. SYTEM OVERVIEW

The proposed system was developed based on the design framework for a 3D spatial gesture interface. The framework supports both gesture acquisition and the modeling of the physical and expressive characteristics that are unique to an individual gesture [10a]. This chapter describes the hardware that acquires the 3D spatial gesture, the software that recognizes the acquired gesture information with the spatial context information.

3.1 Hardware Overview

Gesture acquisition is performed both in hardware and software components. The key idea is to use both visual and body sensors to get optimal gesture features for robust gesture recognition and support various requests of target applications such as 3D object manipulation and scene navigation. Figure 2 shows an overview of the gesture acquisition system. When a user performs gestures, the system acquires the gestures through body sensors and visual sensors. Different body sensors are integrated into a gesture input device that the user attaches to the body or holds in the hand. The device is equipped with a micro-controller for collecting sensor data and Bluetooth wireless technology for transmitting the measurements to a computer. The current setup uses two machines: one for the acquisition system and one for the application. Necessary data is transmitted via the network between two computers. The acquisition receives data from the input device via Bluetooth and from video cameras via Firewire ports. We use the Java Communications API package to

develop a module to receive the transmitted body sensor data. The acquisition module for visual sensors tracks the positions of LEDs on the captured image frames and computes 3D positions. The acquisition machine runs small functional units called filters that process corresponding sensor data in a specific way (converting it to another data type).

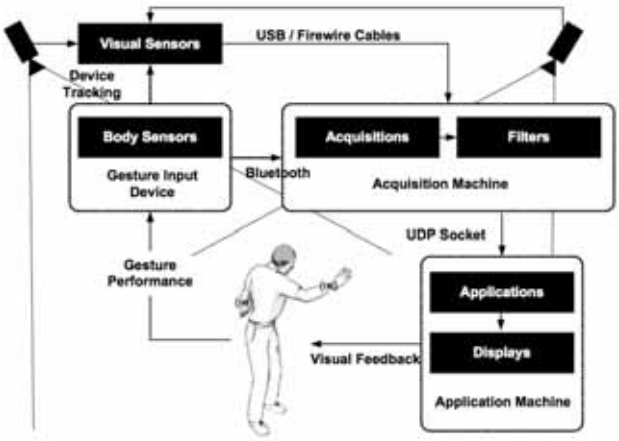

Fig. 2: Overview of the hardware and software framework for gesture acquisition.

3.2 Software Overview

As illustrated in Figure 3, there are two major tasks in the overall framework of our spatial context gesture interface: *spatial context modeling* and *command interpretation*. Acquired sensor data is forwarded from the gesture acquisition module. The spatial context model handles the abstraction and registration of the context data using two types of spatial objects: gesture volume and target. The current status of spatial objects is traced with the performed gestures. The spatial context model provides an efficient way of extracting and retrieving the gesture volume and target. Therefore, system developers can register their own spatial objects and test different configurations in their applications. The command interpretation is accomplished by combining a list of the current context values and the recognized gestures using the gesture model.

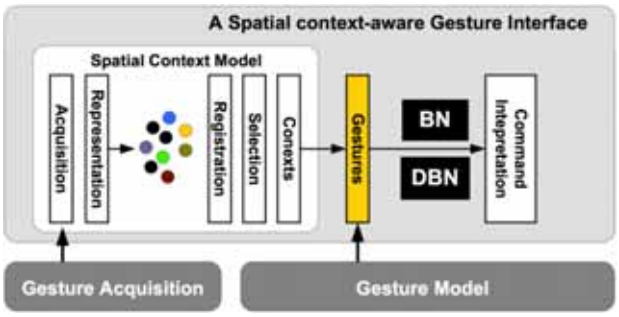

Fig. 3: An overview of a context-aware 3D spatial gesture interface.

4. SPATIAL CONTEXT OBJECTS

In general, the collection of implicit contextual information through available resources is a major challenge in the development of context-aware applications. For the application designer, it is important to decide what context information is relevant to the applications and then test it. The ultimate goal is to improve the usability of the application by optimizing the context information. We focus on the problem of defining the contextual information as it relates to specific gesture information called the spatial information.

4.1 Gesture Targets and Volumes

Spatial information is one of the four aspects of gesture as proposed by Hummels and Stappers [7]. 3D spatial gestures usually relate to objects and locations. The relevance of objects and locations is a key idea in using context information for 3D spatial gesture interfaces. Based on this observation, we defined two types of spatial objects for 3D spatial gesture: gesture targets and gesture volumes.

Gesture targets are physical objects which users can select by moving a gesture input device close to an object or pointing it at an object. They can be physical or virtual objects. Gesture volumes are interaction zones where a set of gestures are planned for the application. Using gesture volumes and targets, we can define gesture candidates during gesture recognition. Therefore, we can minimize the computational complexity by only processing associated gestures within the current contexts. For instance, kitchen space can be defined as gesture volume and the individual components in the kitchen, like a microwave and a refrigerator, can be defined as the gesture target. During the system design, application developers define gesture targets and volumes while considering their design policy about using gestures with the related locations and objects.

4.2 Registration

In our framework, spatial context objects are represented as a three-dimensional Gaussian blob with mean μ and covariance Σ as the center and shape of the object as shown in Figure 3. These Gaussian blobs are modeled and registered based on a series of 3D positions that users provide during the system setup. In Figure 4, a user is registering the 3D position of an object. A user simply locates the device at the position of the object, and registers it by pressing one of the pressure buttons while holding the device on the object's position. The bright color LED of the device allows accurate tracking in various background conditions. We call this direct registration touching. Touching enables users to register the position of the object from the actual position of the object. However, this technique is only available when the object can be reached by the hand inside of the camera volume (i.e. the object should be visible on both camera images).

We use another technique called pointing as proposed in Wilson [11]. This pointing technique allows registration when the target objects are outside of the camera volume or when the users cannot reach the position of the target object. For instance, as shown in Figure 5, when a user wants to register objects on the wall that can't be captured with the current camera setup, then the user can use this pointing technique.

The main purpose is to find the object location by computing the intersection of the multiple pointing rays w_i from different

locations $\mathbf{p}_i$. We can find the covariance of the object μ by solving the linear system of equations via least squares which can be represented as:

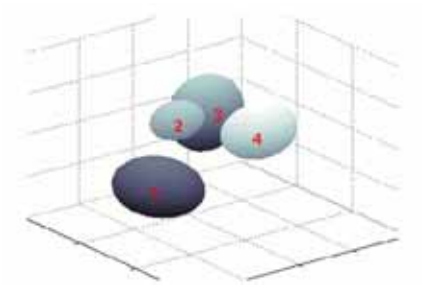

Fig. 3: Representation of spatial objects. Each object is modeled as a 3D Gaussian distribution with mean μ and covariance matrix Σ with a center point c_i, a distance d_i and a variance σ^2

.

Fig. 4: Registering an object using touching. A user is locating a gesture input device (mWire) at the location of the object and registering the positional data to the system by pressing the button of the device.

$$\mathbf{p}_i + s_i\mathbf{w}_i = \mu$$

where $\mathbf{p}_i$ is the device position and $\mathbf{w}_i$ is the pointing ray for the ith pointing observation, and the distances s_i to the object are unknown.

The covariance matrix Σ can be computed by summing up the spread of the differences between calculated target location μ and its estimates $\mathbf{p}_i + s_i\mathbf{d}_i$ and adding some minimal covariance Σ_c i.e.,

$$\Sigma = \Sigma_c + (\mu_i + s_i\mathbf{d}_i - \mu)(\mu_i + s_i\mathbf{d}_i - \mu)^T$$

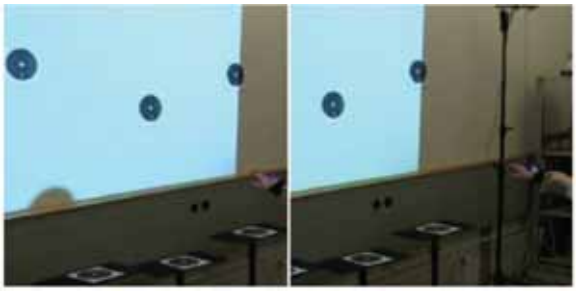

Fig. 5: Registering a spatial object using pointing. A user is performing multiple pointing gestures to the same object from different locations

The pointing direction should be accurate for increased pointing accuracy. Currently, we use the prototype input device by connecting two LEDs (one n the index finger and another on the wrist). Given these two 3D positions, we compute a pointing direction. The accuracy in pointing highly depends on the accuracy of the tracking of the two LED positions. This method is improved using different input mechanisms. For instance, we can use a 6-DOF input device like a 3D mouse that can provide more accurate orientation information even though it is not a wireless device. In addition, we can derive the orientation of the device by combining the digital compass and accelerometers.

4.3 Selection

When the spatial context objects are registered to the system, we then consider the task of object selection. The basic idea is to find objects that are close to the place where users perform 3D spatial gestures. Since each gesture object is modeled as a 3D Gaussian blob as explained earlier, the result of selection is the probability computed from the geometrical relationship between an object and a 3D spatial gesture.

Similar to the registration, there are two different selection techniques: *touching and pointing* depending on whether the object is reached by the input device. As the name suggests, touching selects an object by bringing the input device near the object that the user wishes to interact with.

The system determines which object is closest to the position where the 3D spatial gesture is performed. Each object needs to be modeled as a 3D Gaussian blob with mean μi and covariance Σ_i. A simple technique is to compute the likelihood l_i of selecting object i and evaluating the Gaussian distribution at the point. The likelihood of pointing at target i is

$$l_i = g(\mathbf{p}, \mu_i, \Sigma_i)$$

where $g(x,\mu,\Sigma)$ is the probability density function of the multivariate normal distribution, $\mathbf{p}$ is the 3D position of the device.

For pointing selection, we evaluate the Gaussian distribution at the point that is the same distance away as the hand is from the target and along the ray cast by the hand. The likelihood of pointing at an object i is given by:

$$l_i = g(\mathbf{p} + |\mu_i - \mathbf{p}|\mathbf{d}, \mu_i, \Sigma_i)$$

where p is the position of the hand and w is the ray along the hand.

4.4 System Action Interpretation using BN

In the previous section, we defined spatial context objects (gesture volumes and targets) and described how they can be registered and selected based on the geometrical relationships between the gesture and various objects over various distances. Using this spatial information, we make a better interpretation of the gesture meaning in case it is ambiguous. Now, we explain our system action interpretation technique by combining the selected spatial objects with the gesture recognition results.

To integrate different information statistically, we can create Bayesian network (BNs) that provides a powerful tool for dealing with uncertainty. They enable efficient representation and manipulation of conditional probability distributions for a large number of variables [8].

A Bayesian network is a directed acyclic graph that describes the dependencies among random variables. Each node of the graph represents a random variable. The directed edges define the conditional dependency relations among these variables. This graph structure allows modular representation of knowledge, as well as local, distributed algorithms for inference and learning and facilitates modeling using the intuitive and possibly causal interpretation.

Figure 6 illustrates the Bayesian network used to interpret system actions. The dependencies are modeled with discrete nodes between a system action, a gesture, spatial contexts, and additional contexts. Arrows between the nodes indicate the causal relationship between the action and attributes. The two main attributes are a type of the recognized gesture, and selected targets. Additional context information, time and system functions, can be integrated for minimizing misinterpretations.

In particular, our network uses temporal integration based on the concept of dynamic Bayes network (DBN), a special type of Bayesian networks for time-series events. Depending on the application scenario and user, there are a certain sequence of commands that the user must follow to complete tasks. Therefore, the next command is partly predictable by the previous command history. For example, in a file-operation scenario, the probability of doing the past action becomes much higher when copy or cut is executed recently. Our recognition engine incorporates this prediction information to recognize the final command. The reliability of the gesture recognition engine is improved with this additional information.

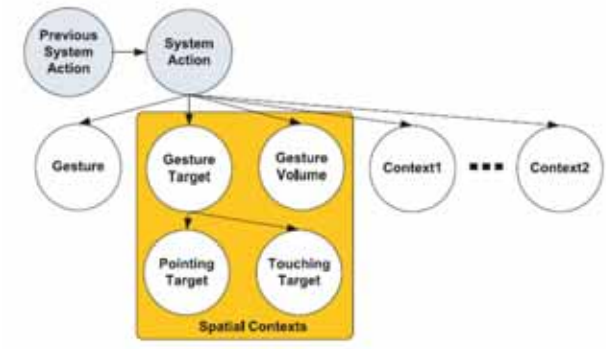

Fig. 6: Topology of the DBN for command recognition. A topology is the predefined connecting of different nodes namely gesture classes, gesture targets and gesture volumes.

For better understanding the use of network in gesture interface, we created make a simple example in a smart environment application. In the application scenario, the interpreted system action is equivalent to a certain command that runs on the system. For example the action "show weather forecast" can be executed as a command when a user is pointing at a window target and performing the *show* gesture. The result is used to execute the command that shows the weather information on the system's display screen.

When the user is pointing at the light, the PointingTarget variable in the Bayes net is set to Light, for example. This causes the Command node to assign equal probability to the "TurnOnLight" and "TurnOffLight" variable settings, since these are the only admissible commands on lights. When the user performs "turn on", the gesture node is set to "TurnOn" and the distribution over the A node collapses to "TurnOnLight". The system then uses the appropriate command to turn on the light.

Fig. 7: Gesture-based interactions in a smart museum environment. (Left) Exhibition items and visual information projected on the wall. (Right)

5. APPLICATIONS

We developed two applications based on the concept of smart environments [3]. We demonstrate the effectiveness of using spatial context information in reducing recognition error and disambiguating 3D spatial gesture performances.

5.1 A Smart Museum Environment

In this application, we developed a smart museum environment where users can interact with physical exhibition items to get related additional audio and visual digital information (Figure 7). For example, users can point at a certain item and listen to an audio description by performing a "turning on" gesture.

For gesture acquisition, we used two cameras installed in positions that efficiently cover the location and orientation of users. During the system setup, the location of each exhibit item is modeled as a 3D Gaussian blob. To train the Gaussian distribution of an item, a series of 3D positions of the LEDs was collected when the pressure button was pressed at the items's location.

The system selects an exhibit item randomly when the hand is close to the item or is pointing at it. After successful gesture recognition, the system provides audio-visual feedbacks such as music or photos associated with the selected target item.

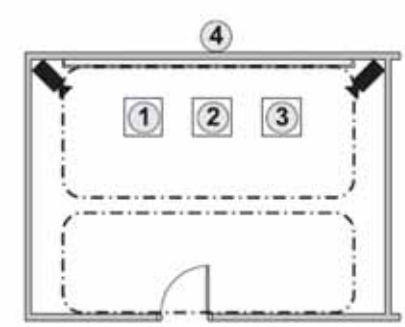

Fig. 8: The setup of the smart museum environment. Number 1 -3 are the targets for exhibit items and the target number 4 is on the wall. Two gesture volumes exist, one for the entrance area and one for the exhibit items respectively.

Figure 8 shows the prototype setup of the smart museum. There are four gesture targets: three (number 1, 2, 3) for exhibit items and one for wall objects (number 4). There are two gesture volumes: one around the entrance area.

The typical user audience is the public. It means that this application is used for a relatively short period. Therefore, we designed the gestures so that users can easily learn and perform them without errors. We used a total of 10 system commands (turning on/off audio and video information), 4 gestures (on/off and up/down), and 4 gesture targets for the exhibit items, and 2 gesture volumes for the entrance and exhibition areas.

5.2 A Smart Home Environment

We designed a smart home environment where users can execute system commands with 3D spatial gestures. We show how the proposed context-aware gesture interface is applied to control home electronics (such as TV, audio system, and lights) in a living room environment.

Figure 9 shows the living room layout used in our application scenario. There are three gesture volumes, *entrance, sofa, and bed*, designed with the main functional areas of the living room environment. When the user enters the room then the entrance volume is activated, the sofa volume is used when the user is listening to music on the sofa. The bed volume is mainly used for sleeping at night. These examples clearly show how gesture volume is related to time and action.

We used five gesture targets: (1) ceiling light, (2) alarm clock by the bed, (3) coffee maker, (4) audio system, and (5) a bookshelf. They are located in the appropriate position for their use in the designed architectural plan. Since we do not have the actual devices, tagged paper boxes represented the devices and their operations were simulated on the computer.

Currently, there are 10 gesture based commands mainly for controlling the gesture targets such as switching on the lights and alarm clocks, and operating the audio system. These are registered as a gesture command which is a combination of a gesture and a target. These commands can be activated by pointing at or touching the gesture targets.

Different from the previous museum application, the typical user of this application is a private user and the system can be customized to a particular user. Therefore, we use gesture registration of our framework to prepare user-designed gesture sets. Typically when the system starts, users are asked to perform a single gesture for each command to setup their own gesture templates.

Once this short initialization process is finished, the application recognizes the new input gestures by comparing them to the same-user gesture sets, and evaluates the performance in relation to reference gesture sets trained by another user for instruction. When the gesture is recognized, its associated command is executed. Obviously, if the user knows how to perform the required 3D spatial gestures, the gesture registration and evaluation process can be skipped. The DTW recognizer can be switched to the HMM recognizer if enough training data is available.

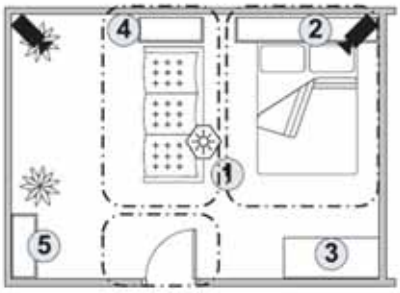

Fig. 9: An experimental setup for a smart home environment. There are four gesture volumes (entrance volume, sofa volume, and bed volume) and five gesture targets (ceiling light, alarm clock, coffee maker, audio system, and a bookshelf).

In addition to these spatial contexts, we used two other context information: typical using time and previous commands. For instance, the user is more likely to turn on the ceiling light at night, and the command to turn on the TV will logically be followed by turning off the TV.

Table 1 shows the suggested sequence of gesture commands using the time and gesture volume information. An example is when a person enters the room, first the user switches on the light followed the audio system. While various different cases could be acquired, we applied a specific architectural knowledge to the various target users. Based on this information, conditional probability for the context information is estimated with the context usages and commands.

During the development of this application, we asked two subjects to use 3D spatial gestures to control the devices in the living room environment described above. They were asked to perform all tasks using the 3D spatial gestures with gesture targets and volumes. During these tests, every 3D spatial gesture performed was transcribed together with the recognition hypotheses, context information at the time of gesture performance and the user's intended command.

We evaluated the accuracy of the baseline system and compared with the one that does not use context information at all. Initially, there were eight gesture recognition errors, two target selection errors and four commands errors. By associating the gestures with spatial objects (gesture targets and volumes) and additional contexts (daytime and previous commands), we resolved gesture recognition errors.

Table 1: A list of command sequences with the used time and the used gesture volume.

ID	Command Sequence	Used Time	Used Gesture Volume
1	4,1,6,8,10	morning(1)	bed(2)
2	7,2,5	morning(1)	door(1)
3	1,6,10	evening(1)	door(1)
4	9	evening(1)	bed(2)
6	7,3,2	evening(1)	sofa(3)

The most prominent phenomenon was that the type of 3D spatial gestures could be misclassified in the gesture recognizer. However, we could eliminate these errors using the spatial context information that defines the relationship between the gestures and the gesture volumes and targets. For example, once the gesture target is selected, we could filter out some of gestures that are not related to the selected gesture target. First the evidences of all other nodes but the gesture node are computed and entered into the network. By inferring the gesture node, we eliminated the gestures with zero or very small probabilities and only compute the likelihood for the other gestures. We also checked the effects of each of the context used. In particular, we analyzed how command recognition errors are reduced using a particular set of contexts. Even though our experiments were for specific tasks, it shows how command interpretation improves by incorporating additional contexts.

6. CONCLUSION AND FUTURE WORK

The paper presents a context-aware 3D spatial gesture interface. Spatial contexts are presented as a specific context for 3D spatial gesture. The paper presented the methods to register and select the spatial contexts, to recognize the final system action. Two prototype applications demonstrated how the proposed interface can be used in real life: the smart home environment and smart museum environment. Each application showed how users can interact with various objects using 3D spatial gestures in digitally augmented environments.

A dynamic Bayesian network for context aware command recognition can be useful to further improve the recognition rate and allow references to objects. This kind of network could be applied in many scenarios, such as an intelligent home, Both improvements of gesture recognition as well as command recognition are possible. For instance better features for gesture recognition or more accurate pointing with a compass sensor can be used instead of two LEDs. A further extension of gesture recognition would be to recognize sequences of gestures with connected HMMs and to support online adaptive learning.

ACKNOWLEDGMENT

This research was supported by Seoul R&BD Program(PA090701)

REFERENCES

[1] Stephen Brewster, Joanna Lumsden, Marek Bell, Malcolm Hall, and Stuart Tasker. (2003). Multimodal ‘eyes-free’ interaction techniques for wearable devices. In CHI ‘03’: *Proceedings of the SIGGCHI conference on Human factors in computing systems*, pages 473-480, New York, NY, USA, ACM Press.

[2] Richard A. Bolt. (1980).Put-that-there: Voice and gesture at the graphics interface. In SIGGRAPH ’80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques, pages 262–270, New York, NY, USA, ACM Press.

[3] Diane Cook and Sajal Das. (2004). Smart environments: Technology, protocols and applications.

[4] Jeremy R. Cooperstock, Sidney S. Fels, William Buxton, and Kenneth C. Smith. (1997). Reactive environments. Commun. ACM, 40(9): 65-73

[5] Guanling Chen and David Kotz. (2000).A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA.

[6] Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. (2004.) a cappella: programming by demonstration of context-aware applications. In CHI ’04: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 33–40, New York, NY, USA, ACM Press.

[7] C. Hummels and P. J. Stappers. (1998). Meaningful gestures for human computer interaction: Beyond hand postures. In FG ’98: Proceedings of the 3rd. International Conference on Face & Gesture Recognition, page 591,Washington, DC, USA, IEEE Computer Society.

[8] David Heckerman, Dan Geiger, and David Maxwell Chickering. (1994).Learning bayesian networks: The combination of knowledge and statistical data. In KDD Workshop, pages 85–96

[9] Wan-rong Jih, Jane Yung-jen Hsu Hsu, and Tse-Ming Tsai. (2006). Context-aware service integration for elderly care in a smart environment. In Modeling and Retrieval of Context Retrieval of Context: Papers from the AAAI Workshop, number WS-06-12, pages 44– 48, Boston, Massachusetts, USA, July 16 –20.

[10] Doo Young Kwon, Markus Gross, A Framework for 3D Spatial Gesture Design and Modeling Using a Wearable Input Device, Proceedings of the 11th IEEE International Symposium on Wearable Computers, 2007, pp. 95-101

[11] A. Wilson and S. Shafer. (2003). Xwand: Ui for intelligent spaces. In Proceedings of ACM CHI Conference on Human Factors in Computing Systems, pages 545–522.

Doo Young Kwon

He received Ph.D in Computer Science at the Swiss Federal Institute of Technology (ETH) Zurich, Switzerland. He received his M.S. degree in Design Computing from Design Machine Group (DMG), University of Washington, Seattle in 2003 and B.S. in Architecture from Ajou University. His research focuses on design computing, spatial media computing, wearable computing.

Ki Tae Bae

He received the M.S. degree in Computer Engineering and the Ph.D degree in Computer Information Engineering from Chonnam National University, Korea in 1999 and 2006, respectively. He is an associate professor of the Department of New Media in Korean German Institute of Technology from 2009. His main research includes Computer Vision, Computer Graphics, Gesture recognition, Image Processing and Image Registration and Media Processing.