CUBOD: A Customized Body Gesture Design Tool for End Users

Jeff K.T. Tang
Caritas Institute of Higher Education &
JST ERATO Igarashi Design Interface Project
Tokyo, Japan
jeff.tang@designinterface.jp

Takeo Igarashi
JST ERATO Igarashi Design Interface Project &
The University of Tokyo
Tokyo, Japan
takeo@acm.org

As motion sensors have become more advanced, gesture-control systems have become more popular in gaming and everyday appliances. However, in existing systems, gestures are predefined by designers or pattern-recognition experts. Such predefined gestures can be inconvenient for specific users in specific environments. Hence, it would be useful to provide end users the flexibility to design and customize gestures to satisfy their own needs. In this paper, we present a system that allows end users to design and customize gestures interactively. A key challenge is that arbitrary user-defined gestures can be difficult for the computer to recognize reliably. A gesture may be too similar to frequent unintentional moves, too difficult to distinguish from other gestures, and/or too difficult to perform consistently. Hence, our system first evaluates the user-defined gesture and then gives feedback on its appropriateness to guide the user in the design of appropriate gestures. A user study demonstrated that users were able to design more appropriate gestures with such guidance than without it.

Interaction Design, Full-body Gestures, Smart Interface

1. INTRODUCTION

In recent years, motion gestures have become a natural user interface (NUI) in various applications. Game devices such as the Microsoft Xbox (2012) equipped with KINECT depth sensors that allows players to use body movement as a game control. Advanced television set like the Samsung Smart TV (2012) could track a user's hand movement as a mouse pointer for menu selection.

However, in existing systems, the gestures to be recognized are predefined by the programmers or designers. These predefined gestures are designed for generic users, sometimes they are inconvenient for users of special needs. For example, a user who hurt his/her the left arm cannot perform a left-hand waving gesture. Besides, it is hard to design generic gestures. Complicated gestures could be distinctive but they are too difficult to memorize; on the other hand, simple gestures are very similar to everyday movements and they may lead to lots of false alarms.

In this paper, we are motivated to address these problems. Our proposed system guides the end users to create good and usable customized gestures quickly. It evaluates every newly created gesture and shows different qualities of a gesture

with a *Radar Chart*. In the meantime, it makes sure the end users are able to repeat the created gestures consistently. With this kind of feedback, they could know what problems a gesture has. Our system can be applied in lecture theatres, surgery rooms, and smart homes (for cooking or web browsing), etc.

The system consists of three stages: *Creation*, *Testing*, and *Review*. In the Creation stage, a new class of gestures is created and samples of this class are recorded. The Testing stage simulates the effect that will occur when a particular gesture is recognized. Finally, the Review stage replays a created gesture and allows the user to practice it until s/he can perform it well. A prototype of our system was developed using Microsoft KINECT SDK. Our major contributions are as follows:

1. We present a gesture-design system consisting of three stages (creation, testing, and review) to efficiently guide gesture design by an end user.
2. Our system provides crucial feedback on new gestures to ensure an appropriate design. Specifically, we present methods for evaluating intra-class consistency, between-class differentiability, and false-positive differentiability

of user-defined gestures. Previous works in this direction such as Ashbrook and Starner (2010) and Kohlsdorf et al. (2011) were designed for knowledgeable users and presented low-level sensor data. In contrast, our system is designed for end users without technical background and provides easy-to-understand visualization and feedback.

3. We implemented a working proof-of-concept gesture-design and -recognition system using the Microsoft KINECT sensors and demonstrated its effectiveness via a user study.

The remainder of this paper is organized as follows. First, we provide a review of related literature. Then, we introduce our proposed system and describe its two major parts, namely gesture design and gesture recognition, in terms of interactions and implementation. This is followed by a detailed description of the calculations involved in gesture design itself (the formulations that allow users to create and test their own customized gestures). Finally, we describe a user study that we conducted to test the system and discuss its results.

2. RELATED WORK

The two major kinds of gestures used in HCI are surface and motion gestures. The former involves the movement of a finger or stylus on a touch screen/pad, and the latter involves movement detected by motion sensors.

2.1. Gesture Controls

The major types of motion gesture include tilt, acceleration, orientation, and full/partial body movements. Gestures can be used to interact with a user interface, particularly an NUI. Tilting is the simplest type of gesture control. Rekimoto (1996) proposed the use of tilt motions to interact with virtual objects such as scroll bars, and Hinckley et al. (2000) demonstrated the use of whole-screen tilting, which has been widely applied in smartphones and tablets.

Gyroscopes and accelerometers can detect orientations and forces, respectively, in three-dimensional (3D) spaces. They are used in game controllers such as Nintendo Wii, smartphones, and robots. However, such devices can only provide information on the relative change in orientation/forces, not absolute coordinates, limiting their application.

In contrast, motion gesture systems can track human movement accurately but until recently have been very expensive. However, the relatively recent development of the Microsoft KINECT device has changed that situation, providing full-body motion

sensing at an affordable price. Programmers have begun to develop home automation systems using the KINECT depth sensors; for example, Forss (2011) reported using the sensor to detect the location of a human in a room and provide that area of the room with suitable illumination. Gallo et al. (2011) proposed a system for medical doctors through which scanned medical images can be manipulated using hand gestures to avoid touching documents during surgical operations.

2.2. Motion Gesture Design

There are two general approaches to motion gesture design: end user elicitation and demonstration.

2.2.1. Elicitation

In this method, end users are encouraged to create motion gestures by showing them visual clues, which are used by Wobbrock et al. (2005) for surface-gesture design via so-called guessability studies through which a consensus among user-defined gestures is determined based on the decisions of a majority of users. Wobbrock et al. (2009) developed a *touch table* (a large touch screen oriented horizontally) on which various UI operations such as dragging, enlarging, and moving objects could be performed; they had volunteers use the system and then used their set of gestures to improve the design of the system. Ruiz et al. (2011) applied a similar approach to analyze user-inspired smartphone motion gestures.

2.2.2. Demonstration

In this method, a designer provides samples that are recognized by the system. Long (2011) developed *Quill* that is an interactive design tool for pen gestures. The designer creates a gesture, and Quill provides feedback on the gesture (e.g., about its quality) that aims to help the designer to improve the gesture. However, such systems can be difficult to understand, and they do not test whether the input gesture conflicts with the existing gesture classes, which is difficult for any users (even expert users) to determine. Also, the user cannot test the created gesture, so its performance remains unknown.

Crayons proposed by Fails and Olsen (2003) is an interactive tool for training a vision-based classifier. Pre-recorded images are input into the system as samples. The user can interactively tag the images as various classes, explicitly providing immediate feedback on system performance. However, it is too technical for end users.

Newer gesture-design tools have adopted a three-stage design-test-analysis model, which is more comprehensive and helpful for iterative gesture design. SUEDE proposed by Klemmer et al. (2000)

(a)

(b)

Figure 1: (a) The setup of our smart home environment; (b) The architecture of CUBOD, showing how everyday appliances are connected to the system.

is a prototyping tool for speech user interfaces that incorporates this model. In the design stage, the designer creates samples of conversation. In the test stage, testers use the samples. In the analysis stage, the system analyzes the data collected during the testing stage and uses the results to improve the next iteration of design.

Hartmann et al. (2007) developed *Exemplar* that assists in fast prototyping of gesture interactions. The designer designs a gesture, and then the system is connected to hardware for testing such as the *d.tool* proposed in Hartmann et al. (2006). After testing and analysis, the user can directly edit the gesture and tune the thresholds in the recognizer. However, this system does not consider consistency among trials and distinctiveness relative to other gesture classes, and therefore it cannot ensure that designed gestures will not interfere with other ones. Furthermore, it does not provide a way to manage many gesture classes/samples.

MAGIC proposed by Ashbrook and Starner (2010) is a more generalized tool for creating sensor-based interactive systems. It is similar to *Exemplar*, but it considers information about consistency between samples, the chances of getting false-positives during recognition, and the distinctiveness of classes of gestures, all of which is used to improve the recognition rate of the motion gestures created by designers. It has other beneficial aspects, but a user study indicated that many users had difficulty understanding how to use the system. Kohlsdorf et al. (2011) enhanced MAGIC by filtering false-positive gestures using iSAX developed by Shieh and Keogh (2008). iSAX and Continuous Dynamic Programming by Yaguchi et al. (2008) are state-of-art fast techniques for matching a gesture against a large everyday-gesture database. However, existing tools are designed for prototyping systems with

wearable motion sensors, so it allows users to perform low-level operations like connecting different sensors, view raw motion signals and fine-tuning the classifier thresholds. As a result, these systems are targeted for technical users, such as designers, but not for end users.

In contrast, our proposed system is designed for both technical users and non-technical users. It is implemented with KINECT that does not required to wear sensors on the body. Instead of raw signals, we interpret the goodness of each designed gesture in high-level concepts, which could be understood easily. Kray et al. (2010) determined the appropriateness of user-defined gestures with mobile phones through a user test, but it is not determined automatically.

In addition, since the existing tools are for prototyping so the created gestures cannot be modified by the users except the designer him/herself. Williamson and Murray-Smith (2012) developed a reinforcement learning method that encouraged people to design original gestures using audio feedback. Our system allows users to design original gestures according to their own style, and can be changed time by time. Graphical and audio feedbacks are provided as guidance in the gesture-design process.

3. THE PROPOSED CUBOD SYSTEM

The proposed system called the Customized Body Gesture Design System, abbreviated as *CUBOD*. It consists of two modes: *gesture design* and *gesture recognition*. The user defines new gestures in design mode and then tests it in recognition mode. Figure 1 shows the architecture of our system applied to a smart home, which consists of a room with an electric fan, two lights (left and right), an LCD picture frame, and a TV set. The on/off switch of the fan

is controlled remotely by infra-red signal; the on/off and brightness of the two lights are controlled by a dimmer box; the on/off and forward/backward picture functions of the photo frame are directly controlled by a PC; and the TV monitor on/off, play/stop, and next/previous movie functions are also controlled by the PC. A KINECT device acquires motion gestures performed by a user. The system monitors the motion stream, recognizes gestures, and invokes the function mapped to a recognized gesture. Note that the main interface is displayed on a computer monitor. The smartphone is treated as an auxiliary interface for switching to different modes.

3.1. Data Acquisition

The KINECT device consists of a RGB camera and depth sensors, so it is able to capture an RGB-D video stream (where D is the depth dimension) along with color frames. The latest version is able to record the movements of up to four players in real time. The Microsoft KINECT SDK tracks the human body, returning 20 joints per frame at a frame rate of about 30 frames per second (fps). The joint hierarchy is shown in Figure 2(a).

Figure 2: (a) The hierarchy of human body joints obtainable by the Microsoft KINECT SDK; (b) The five body parts we considered in the feature extraction.

3.2. Motion Feature Extraction and Gesture Recognition

We group joints z into five body parts, including four sets of limb joints: *left arm*, *left leg*, *right arm* and *right leg* (denoted by $J_L = \{z_{L,k} | k = 1, 2, \dots, K\}$ where K is the size of the limb set), and a set of torso joints (denoted by $J_T = \{z_{T,l} | l = 1, 2, \dots, L\}$ where L is the size of torso set). In the present study, the end-effectors were not considered because they tend to show large variation, and they are often not tracked well. In the motion feature, we consider the movement of each limb relative to the torso.

We compute the feature vector F based on the weighted Euclidean distance between two joints on different bones of a skeleton (Tang et al. (2008)). As it measures the joints' relative movements, there is no need to normalize user's gesture to the same

origin and facing direction. Each feature f in the vector F is the average distances between the sets of limb and torso joints:

$$f_{k,l}(J_L, J_T) = \|z_{L,k} - z_{T,l}\| \quad (1)$$

Suppose we have two motion gestures: $M_1 = P_1, \dots, P_q, \dots, P_Q$ and $M_2 = P_1, \dots, P_r, \dots, P_R$, where Q and R are their number of frames. The posture-level distance score between two postures P_q and P_r is hence given as follows:

$$d_P(P_q, P_r) = \sum_{k=1}^{K} \sum_{l=1}^{L} D_{l,k}^{P_q, P_r} \quad (2)$$

Where $D_{l,k}^{P_q, P_r} = \|f_{k,l}(J_L^{P_q}, J_T^{P_q}) - f_{k,l}(J_L^{P_r}, J_T^{P_r})\|^2$. Then, the distance between any two gestures $d(*, *)$ is calculated by aligning the sequences of postures by Dynamic Time Warping (DTW).

3.3. Identification of Stationary Gestures

We segment the motion stream into primitive movements by detecting the stationary posture, i.e., a posture at which all joints have zero acceleration. In the motion stream, we apply a sliding window over time (with overlap) and acquire a range of 30 frames for stationary gesture analysis instead of using the posture, thereby avoiding over-segmentation.

3.4. Identification of Stationary Gestures

To lower the chance of getting false-positives in the recognizer, we record everyday gestures that are not already included in the subset of control gestures. The everyday-gesture dataset includes a variety of movements within the room, for example, walking around, reading a book, making a phone call, sleeping on a sofa, and so forth. We use this dataset to provide feedback to the user in the design phase, which we describe in the next section. We segment the gestures in the motion stream at stationary postures (i.e. average acceleration near to zero).

4. GESTURE DESIGN MODE

The gesture-design process consists of three stages (creation, testing, review) inspired by existing gesture prototyping tools such as MAGIC by Ashbrook and Starner (2010), Exemplar by Hartmann et al. (2007), and SUEDE by Klemmer et al. (2000).

The user first creates a gesture class for a particular function of a device and then records samples. A

gesture class represents one type of movement, such as "swiping the right hand to the right," which is mapped to a function such as "Go to the next channel." Because any given user will show a degree of variation among his or her attempts to carry out a gesture, the user is required to perform each gesture several times; each recorded attempt is considered one sample.

Second, the user tests the created gestures on the system, which shows the recognition results, informing the user how well the gesture recognizer works with these gestures. Then, the user can modify gestures accordingly. Additionally, the user can review stored gestures.

The workflow of the gesture-design tool is shown in Figure 3. In order to achieve a higher usability, the number of options at every stage has to be minimized. To simplify the interaction, we only give users two options to choose from at a time. For example, if a user wishes to select the default setting, s/he does not have to do anything. On the other hand, If the user wishes to select the other option, s/he has to do something like waving his/her hand.

Figure 3: The workflow of the gesture-design tool in CUBOD.

The system forces the user to go through all necessary steps. If anything is skipped, the system prompts the user to address the problem. Thus, our system guides the user in the creation of gestures; the user does not even need to decide what function to choose. This dramatically simplifies the gesture-design process, making it usable for even novice users.

4.1. Gesture-creation Stage

The user invokes the gesture design mode through the auxiliary smartphone interface. Figure 4 shows how the user inputs the name of a gesture class (we use the device/function pair to identify a

gesture class), a device, and its function using the smartphone.

Figure 4: The smartphone is used as an auxiliary device for select (a) a device, and (b) a function of the selected device, for identifying the gesture to be designed.

The design of the gesture-creation interface is shown in Figure 5 (on next page). When the user creates a new gesture, s/he has to stand in front of the KINECT that connected to a nearby TV set or dedicated computer monitor. After inputting the gesture class via the smartphone, the system automatically invokes the gesture-creation stage, as shown in Figure 5(a), and then the user is required to demonstrate the desired gesture three times. A goodness value is assigned to the input gesture samples based on consistency over all trials (S_C), distinctiveness compared with other gesture classes (S_G), and distinctiveness compared with everyday gestures (i.e., robustness against accidental triggers, S_A). Real values are used to represent these qualities, and these are plotted to a Radar Chart. Finally, the performance is classified into one of three levels: good, average, and bad. The detailed calculations involved in this process are described in the next subsection.

Goodness values (S_C , S_G , and S_A) are presented in a radar chart, as shown in Figure 5(b). A radar chart is used for organizing the characteristics of a single topic, where each point of the radar lists a fact, attribute, or trait about the topic Chambers et al. (1983). Here, each attribute represents a quality, each of which has four levels represented by concentric circles.

4.1.1. Consistency of Input Samples (S_C)

This measures the similarity between all input gesture samples. Higher similarity values indicate smaller variation between samples, which ensures that the created gesture can be repeated easily in appliance-control mode.

Figure 5: The gesture-creation interface. (a) A user recording a gesture sample. (b) The system assigns goodness values to input gesture samples. A count-down timer is on for choice confirmation. (c) Based on the results, the user can redesign the same gesture or choose another one. (d) The user can use the smartphone interface to make adjustments.

Let I be the set of input samples and a gesture M_I^i be the i th sample in I . We calculate the difference between each sample pair (i, j) where $i \in I, j \in I$ and $i \neq j$, with the distance function $d(*, *)$ defined previously. Let $|I|$ be the cardinality of set I . Then, the equation is given by Equation 3.

$$V_C = \left\{ \frac{|I|(|I| - 1)}{2} \right\}^{-1} \left\{ \sum_{i \in I} \sum_{j \in I, i \neq j} d(M_I^i, M_I^j) \right\} \quad (3)$$

With the geometric arctangent ($\text{atan}()$) function, we are able to map the values of V_C , which are real numbers, into a finite range between $[0, 1]$, since $\tan(0) = 1$ and $\tan(\pi/2) = \infty$. However, because a large value represents low similarity, we consider a complement to 1 for V_C . Hence,

$$S_C = \left\{ \frac{\pi}{2} - \tan^{-1} \frac{V_C}{\pi/2} \right\} \quad (4)$$

4.1.2. Distinctiveness Compared with Other Gesture Classes (S_G)

The greater the S_G value is, the smaller is the chance that this gesture will be misrecognized as a different gesture. Again taking I , and D as the sets of input samples and the samples of an existing gesture class in database respectively, let Z be the set of all gesture classes and $M_{D,k}^i$ be the i th sample in the k th gesture class in Z , we determine the maximum difference between each input gesture-database gesture pair. The equation is given by:

$$V_G = \max_k \left\{ \frac{1}{|S||D|} \sum_{i \in S} \sum_{j \in D} d(M_I^i, M_{D,k}^j) \right\} \quad (5)$$

Then, S_G can be normalized to a range $[0, 1]$ by an arctangent function. Unlike S_C , it is a measurement of difference, and there is no need to consider the complement value. Hence, we have Equation 6.

Figure 6: The gesture-testing interface. (a) The user is performing a gesture to test the recognition result. (b) The user can input a gesture name using a smartphone. If the gesture does not exist, then the system prompts the user to create it.

$$S_G = \tan^{-1} \left\{ \frac{V_G}{\pi/2} \right\} \quad (6)$$

4.1.3. Robustness to Accidental Triggers (S_A)

This measures the differences between input samples and everyday gestures, which are recorded by the system in recognition mode. The greater the S_A value, the lower the chance of having false positives. Let E be the set of everyday gestures and M_E^k be the k th sample. The equation is given by:

$$V_A = \max_k \left\{ \frac{1}{|S|} \sum_{i \in S} d(M_T^i, M_E^j) \right\} \quad (7)$$

Similar to the calculation of S_G , the value of V_A is then normalized by:

$$S_A = \tan^{-1} \left\{ \frac{V_A}{\pi/2} \right\} \quad (8)$$

4.1.4. Goodness Score (G)

The goodness score G is derived from the above three qualities as shown in Equation 9. The default value of each threshold is set to 0.5, and the input gesture class is automatically classified as good, average, and bad when G is 3, 2, and 1, respectively. This interface provides the user with a clear graphical representation of his/her performance. Using this information, the user can decide to add this gesture to the database or redesign it.

$$G = \sum (S_C > T_C, S_G > T_G, S_A > T_A) \quad (9)$$

When the user redesigns a gesture, the three thresholds T_C , T_G , and T_A can be readjusted through the smartphone interface, as shown in Figure 5(d). In the system, we use the term *importance* instead of *threshold* so as not to confuse nontechnical end users. When the importance (threshold) value is increased, a higher score is needed to get an evaluation of good, and vice versa.

4.2. Gesture-testing Stage

In the gesture-testing stage, the user performs different gestures in recognition mode and can change the classifier threshold to improve the gesture. The interface is shown in Figure 6. The real-time RGB video of the user's body is shown on the left side, which allows the user to see what s/he is doing and to make sure that body movements are properly tracked by the KINECT depth sensors. The virtual room is shown on the right side, which animates the corresponding effect when a gesture is recognized.

For example, when the user swipes the right hand up, which is assigned to making the right light brighter, the right light in the virtual room is animated to look brighter. In this way, the user can easily confirm that the gesture was correctly recognized.

The user can also trigger an effect by selecting a device/function pair using a smartphone. If there is no gesture for this pair, the system will prompt the user to create one. In this case, a binary option will pop up. If the user does not move, the system will automatically select the default choice of *create this gesture*.

Figure 7: The gesture-review interface. (a) The user can choose a gesture to review and practice. (b) If the performance is not good, the system prompts the user to redesign it.

4.3. Gesture-review Stage

In this stage, the user can choose a gesture to review and practice. This stage helps users to remember gestures and the functions assigned to them. Figure 7 shows the interface of the gesture-review stage.

A user can pick a gesture to revise by selecting a device/function pair with a smartphone. As in the testing stage, if a gesture for the pair has not yet been created, the user will be prompted to design a new gesture. Otherwise, the user can review the selected gesture, choose to review another gesture, or quit the design tool and go back to recognition mode.

When a gesture is reviewed, a video of the selected gesture is played on the right side of the screen. To let the user get familiar with the gesture, the video is played twice in slow motion and then twice at normal speed. During these four trials, the user practices the gesture by mimicking the video, and these movements are recorded by the system. The similarity between the actual gesture and the template is computed, and an evaluation (i.e. Good or Bad) is displayed below the video. If the performance is not good, the gesture is most likely too complicated, and the system will suggest that the user redesigning it.

5. USER EVALUATION

We evaluated our system via a user study that included 30 subjects (20 males and 10 females) with different occupations such as university students, clerks and researchers. More than a half of them are neither from computing nor design disciplines. Most of them have none or a little experience on body sensing device like Wii and KINECT.

5.1. Experiment Setup

The experimental apparatus as shown in Figure 8 was established, and only one subject at a time was allowed to use the system. Each subject used the CUBOD system to create gestures. A facilitator observed aside and a video camera was set up to record the entire test process.

Figure 8: The monitor displays the UI of CUBOD. Each subject uses it to create customized gestures and test them.

The experiment consists of 3 steps. Firstly, the system recorded everyday gestures performed by each subject. He/she was told to perform everyday household tasks such as reading a book, sweeping the floor, moving a box for five minutes. Secondly, each subject was required to design four gestures with CUBOD, which coincide with the commands such as turning on/off the TV, fan, etc. Finally, each subject tried to control the appliances with the gestures created previously. The percentage of successful recognition was recorded. As control experiment, each subject also tried to create gestures without giving any feedback such as the Radar Chart.

Each subject has filled an user evaluation questionnaire as shown in Table 1 to rate their experience in a 9-point scale. A higher score means a subject agree with a statement more.

Table 1: The user evaluation questionnaire.

#	Statements
1	The CUBOD system is useful for tailor-making gestures for your own preference.
2	The Radar Chart lets you know more clearly the quality of each gesture.
3	The feedback information is helpful for gesture creation.
4	The <i>Gesture-practice</i> mode helps you to perform consistent gestures.
5	The <i>Gesture-review</i> mode helps you to remember the gestures.
6	The CUBOD system is easy to use.
7	I am confident that CUBOD can apply to everyday appliances.

5.2. Result and Discussion

Figure 9 shows the average score of the user evaluation questionnaire. The statements 1 to 5 (regarding the proof-of-concept) received higher scores than the statements 6 to 7 (regarding the application). In overall, the subjects felt CUBOD is useful for customizing gestures. In particular, they found that both the *practice* and *review* functions are the highly helpful in gesture creation.

Figure 9: The result of the user evaluation questionnaire. The statements refer to Table 1.

The statements 2 and 3 received a score of 7.3 and 7.1 respectively, which shows that an iterative design with guidance leads to produce better quality gestures, which are suitable to be used as control signals. From the feedback provided by CUBOD (especially the Radar Chart), the subject could know whether his/her design is likely confused with the other existing gestures as well as the everyday gestures.

The statements 6 and 7 received a lower score. We believe that the application has been affected a lot by the quality of depth sensors and the gesture

recognition algorithm. One limitation of KINECT is that it cannot identify fine hand gestures. Therefore, the users can only create gestures with more vigorous moves. Moreover, KINECT can only acquire depth data from single viewpoint, gestures such as clapping hands cannot be estimated well as some joints are missed during tracking. Such kind of noise would heavily affect the gesture recognition process. As a quick fix, we can interpolate and smooth the trajectories of each joint. Now the CUBOD achieved a gesture-recognition accuracy of 90%. However, there still a small amount of accidental triggers by Everyday gestures (about 3%). With simple Dynamic Time Warping (DTW), a standing gesture (both hands put down) is likely conflicting with the hand-swiping up gesture because these gestures shared a lot of similar postures. However, the CUBOD system is a generic solution so we believe that the user experience could be improved by adopting a more robust gesture recognizer.

6. CONCLUSION AND FUTURE DIRECTIONS

Our system, CUBOD, allows end users to design and customize their own motion gestures interactively in a guided way without pressing buttons. Our interface replaces technical information with simple, meaningful graphs that provide information on consistency over all trials, distinctiveness compared with other gesture classes, and distinctiveness compared with everyday gestures. The user can also review and modify any given gesture.

The major contributions of this paper are as follows. We solved the gesture registration problem by considering stationary gestures. We proposed a gesture-design interface that gives users meaningful suggestions and allows them to recover from errors by following simple binary guided operations. We used the KINECT depth sensors to acquire full-body gesture data. We demonstrated our system using a smart home in which appliances could be manipulated via a suite of gestures.

CUBOD is a generalized framework that can be applied to different environments and purposes such as a Lecture Halls, Kitchen, Surgery Rooms, Web browsing, Gaming, etc. In future work, we will provide greater flexibility for users to configure the appliance profile in an organized way. We will also enhance our system to track multiple users at a time, using techniques such as facial recognition.

ACKNOWLEDGEMENT

This work was fully supported by JST ERATO research grant. We thank the anonymous reviewers for their useful comments and suggestions.

REFERENCES

- Ashbrook, D. and Starner, T., (2010) *MAGIC: A Motion Gesture Design Tool*. In Proc. of the 28th Int'l Conf. on Human Factors in Computing Systems, CHI'10, pp.2159-2168.
- Chambers J.M., Cleveland W.S., Kleiner B. and Tukey P.A., (1983) *Graphical Methods for Data Analysis*. Chapman and Hall, New York.
- Fails, J. and Olsen, D., (2003) *A design tool for camera-based interaction*. In Proc. of the SIGCHI Conf. on Human factors in computing systems, CHI'03, pp.449-456.
- Forss, P.K., (2011) *Kilight - Controlling Tellstick with Microsoft KINECT*. <https://www.youtube.com/watch?v=zIChtCHcKJw>.
- Gallo, L., Placitelli, A. and Ciampi, M., (2011) *Controller-free Exploration of Medical Image Data: Experiencing the KINECT*. In Proc. 24th Int'l Symposium on Computer-Based Medical Systems, June 2011, pp.1-6.
- Hartmann, B., Abdulla, L., Mittal, M. and Klemmer, S.R., (2007) *Authoring Sensor-based Interactions by Demonstration with direct Manipulation and Pattern Recognition*. In Proc. of the SIGCHI Conf. on Human factors in computing systems, CHI'07, pp.145-154.
- Hartmann, B., Klemmer, S.R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A. and Gee, J., (2006) *Reflective Physical Prototyping through Integrated Design, Test, and Analysis*. In Proc. of the 19th Annual ACM Symposium on User Interface Software and Technology, UIST'06, pp.299-308.
- Hinckley, K., Pierce, J., Sinclair, M., and Horvitz, E., (2000) *Sensing Techniques for Mobile Interaction*. In Proc. of the 13th Annual ACM Symposium on User Interface Software and Technology, UIST'00, pp.91-100.
- Klemmer, S.R., Sinha, A.K., Chen, J., Landay, J.A., Aboobaker, N. and Wang A., (2000) *SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces*. In Proc. of the 13th Annual ACM Symposium on User Interface Software and Technology, UIST'00, pp.1-10.
- Kohlsdorf, D., Starner, T. and Ashbrook, D., (2011) *Magic 2.0: A web tool for false positive prediction and prevention for gesture recognition systems*. In Proc. of the IEEE Int'l Conf. on Automatic Face and Gesture Recognition and Workshops, FG'11, pp.1-6.
- Kray, C., Nesbitt, D., Dawson, J., and Rohs, M., (2010) *User-defined Gestures for Connecting Mobile Phones, Public Displays, and Tabletops*. In Proc. of the 12th Int'l Conf. on Human Computer Interaction with Mobile Devices and Services, MobileHCI'10, pp.239-248.
- Long, A.C., (2001) *Quill: A Gesture Design Tool for Pen-based User Interfaces*. PhD thesis, Carnegie Mellon University, 2001.
- Microsoft, (2012). *KINECT for Windows - Voice, Movement and Gesture Recognition Technology*. <http://microsoft.com/en-us/kinectforwindows>.
- Rekimoto, J., (1996) Tilting operations for small screen interfaces. In Proc. of the 9th Annual ACM Symposium on User Interface Software and Technology, UIST'96, pp.167-168.
- Ruiz, J., Li, Y., and Lank, E., (2011) *User-defined motion gestures for mobile interaction*. In Proc. of the 2011 Annual Conf. on Human Factors in Computing Systems, CHI'11, pp.197-206.
- Samsung, (2012). Let the Smart TV Experience begin. <http://www.samsung.com/>.
- Shieh, J. and Keogh, E., (2008) *iSAX: indexing and mining terabyte sized time series*. In Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining, KDD'08, pp.623-631.
- Tang, J.K.T., Leung, H., Komura, T. and Shum, H.P.H., (2008) *Emulating human perception of motion similarity*. Computer Animation and Virtual Worlds 19, 3-4 (Sept. 2008), pp.211-221.
- Williamson, J. and Murray-Smith, R., (2012) *Rewarding the original: explorations in joint user-sensor motion spaces*. In Proc. of the 2012 ACM Annual Conf. on Human Factors in Computing Systems, CHI'12, pp.1717-1726.
- Wobbrock, J.O., Aung, H.H., Rothrock, B. and Myers, B.A., (2005) *Maximizing the guessability of symbolic input*. In CHI'05 extended abstracts on Human factors in computing systems, CHI EA'05, pp.1869-1872.
- Wobbrock, J.O., Morris, M.R. and Wilson, A.D., (2009) *User-defined gestures for surface computing*. In Proc. of the 27th Int'l Conf. on Human factors in computing systems, CHI'09, pp.1083-1092.
- Yaguchi, Y., Naruse, K. and Oka, R., (2008). *Fast spotter: An approximation algorithm for continuous dynamic programming*. In Proc. of the 8th IEEE Int'l Conf. on Computer and Information Technology, CIT'08, pp.583-588.