2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing

# Gesture-Based Easy-Computer Interaction using a Linear Array of Low Cost Distance Sensors

Guan Huang and Seng W. Loke
Department of Computer Science and Computer Engineering, La Trobe University
Melbourne, Australia
Email: bosonhuang@gmail.com, s.loke@latrobe.edu.au

*Abstract*— **In this paper, we introduce the concept of Easy Computer Interaction (ECI), and describe our design and implementation of an ECI application prototype using a collection of IR distance sensors. Performance and limitations of the application are analyzed.**

***Keywords-easy-computer interaction; gestures; IR sensors; low cost***

## 1. INTRODUCTION

In Human-Computer Interaction (HCI), improvements on both the computer interface and human input modalities have been explored in order to achieve more efficient and easier interaction [1]. By using Human Interface Devices (HIDs), such as sensors, the interaction space between human and computer can be extended to a wider physical boundary, from a plane surface to three-dimensional space. In addition, easy plug-in use and convenient adaptation of sensors to applications are helpful. In this paper, we introduce a low cost concept of a sensor-based interactive equipment, where low cost means low price, low power consumption and low barriers to usage. Here, we define *Easy-Computer Interaction* (ECI) as a kind of interface that focuses on low cost, ease of use and efficiency of functions.

IR distance sensors can be attached to any physical surface converting the surface into a computer interface, such as a wall, around a box, on a table, etc. As a result of this simple implementation, a user could perform quick and straightforward interactions with the sensors which are connected to a computer. For example, performing a gesture over the arm of an armchair (i.e., over the sensor attached to the arm) could control the volume of an entertainment application or operate Microsoft PowerPoint slide shows.

## II. RELATED WORK

### A. Approaches Using IR Sensors

Kratz and Rohs [2] equipped a mobile device with distance sensing capabilities that aims to allow interaction with the mobile device or wearable devices via gestures. In their prototype, they demonstrated an Apple iPhone equipped with six IR distance sensors, in order to read and recognize coarse hand movement-based gestures.

Pinak Parekh [3] installed a user gesture interactive interface called Dunhill Diary with distance sensing function and image projection. A user could browse Dunhill's history and its luxury stories though a display board. Simple hand movement from right to left is recognized and used to change the slide show on the projected film display.

Ishikawa *et al*. [4] introduced a touchless input device and gesture commands for controlling simple applications on a PC. A distance sensor is used for touchless convenience and avoiding distractions when the user communicates with computers.

Sphere is a multi-touch-sensitive spherical display system enabling an easy 360-degree access for multiple users. It provides functionalities for displaying and scaling pictures, displaying movies, and spherical painting [5].

### B. Approaches Using Other Types of Sensors

Funky Wall [6] is an interactive wall-mounted display system that presents mood boards using gesture, speech and visuals. It helps designers communicate their ideas to the clients through the mood boards that are created or used in the early stages of the design process. ThinSight [7] is a multi-touch optical sensing system. It was fully integrated in a thin form-factor screen, to capture finger pointing and multi-touch detection. WUW [8] is a wearable gestural interface, which attempts to bring information out into the tangible world. Through a wearable camera and projector, a user could interact with the system using gestures and gain information via the projected information on walls, hands, or physical objects. Wang *et al.* [9] presents an interactive game using speech and gesture recognition. User use a wand equipped with an accelerometer to perform gestures in front of a display board. This system allows the user to play games, such as spelling words using wand movement and speech. Lee [10] invented a low-cost multi-point interactive whiteboard using the WiiMote. By using a WiiMote equipped with a projection screen or LCD display, the application can track the IR light based pen, and follows the track to perform writing or painting on the screen or LCD display. MirrorBoard [11] is an interactive billboard that allows users to interact with it, such as switching between advertisements and placing participants' pictures into the screen.

## III. CONCEPTS AND DESIGN

In the operation of our application, the user could perform preset tasks with the computer which is attached with IR sensors. The user interacts with the system through hand movement over a number of sensors and gains responses from a display board or a projector screen. Gestures will be captured and analysed by the application. Successful gestures will be recognized and matched with supported

978-0-7695-4612-4/11 $26.00 © 2011 IEEE
DOI 10.1109/DASC.2011.61

gesture patterns in the system after the user's input events have been analysed and the application will perform corresponding tasks such as displaying particular contents. Gestures design and recognition will be discussed in the following subsections.

### A. Sensors Used

There are several types of IR distance sensors currently available on the market. One of the most widely implemented and comparatively low-priced IR distance sensor is from Sharp. In our design, we used eight GP2D120X model IR distance sensor from Sharp [12].

*Power Consumption and Cost Analysis*

The supply voltage of eight sensors working together during a normal indoor temperature ranges from -0.3 to +7.0 volts, which means any PC or laptop could provide enough power supply through the USB cable to sensors. In our design, we use IR distance sensors connected to a distance sensor adapter (motherboard controlling the IR sensor). Phidgets [14] provides the adapter at a reasonable price with a development kit. The characteristic of low power consumption and inexpensive equipment installation meets our low cost and ease of use goals.

*Reading Performance*

The Sharp 2D120X IR distance sensor offers a valid range of detection from 10cm to 80cm. the sensor produces an analog output value that varies according to the distance of an object from it within the valid detection range. For the ease of programming and recognizing, the produced analog output SensorValue could be translated into Distance. The formula [13] below shows the translation and is only valid for a SensorValue from 80 to 500:.

$$\text{Distance (cm)} = 4800 / (\text{SensorValue} - 20)$$

In addition, the sensors read distances in a particular time slot. The time slot for a single measurement is 38.3ms ± 9.6ms, with a gap between each measurement at maximum of 5.0 ms.

### B. Supported Gestures

In our design, sensors are placed in an array on a horizontal plane like on a table as a line controller. Gestures or movements of user's hand palm are mapped in the following ways. Figure 1 shows the supported gestures.

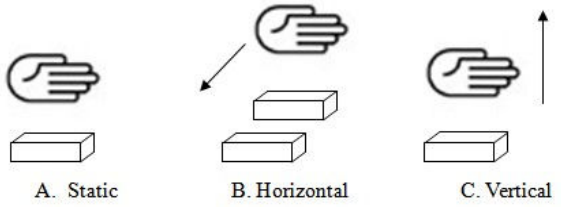


Figure 1: Supported gestures

- Static position-based gesture:
  Palm stays over a single sensor for a particular period of time to perform preset tasks.
- Horizontal palm movement-based gesture:
  Hand palm movement over a number of sensors in one direction from left to right and reverse.
- Vertical palm movement-based gesture:
  Hand palm movement over a single sensor in one direction from up to down and reverse.
- A composition of the above movement-based gestures:
  Hand palm movement over a number of sensors in directions comprising a combination of vertical and horizontal gestures, such as a "box" movement.

Each gesture effectively generates a sequence of sensor input readings, and we regard these sequences of readings as events to the application. Each event is regarded as a sequence of data records sensors information in which we named it "reading sequence", such as sensor number, sensor reading values and time gaps between readings. Distance and time measurements are two aspects that we considered as vital for gesture recognition algorithms that analyse those data sequences. Thus, event fire conditions are the conditions where a data sequence matches preset gesture data patterns. It is calculated in two steps according to distance and time measurements.

### C. Distance Measurement in Event Fire Conditions

The first step is to analyse the distance (of detected hand from sensor) readings of each event change. Distances at a similar level over a single sensor will be considered as a static position-based gesture. Detected distance changes on a single sensor will be considered as a vertical hand palm movement and further analysed to determine whether it is an up-down or down-up movement-based gesture. Distances at a similar level over a number of sensors will be considered as a horizontal hand palm movement and further analysed to determine whether it is a right-left or left-right movement-based gesture. Figure 2 below shows the distance measurement range of each section and valid maximum distances in gesture recognition.

### D. Time Measurement in Event Fire Conditions

The second step is to analyse whether those recognized movements from step one are within a valid reading period of time. We initially set the time gap between each successful readings to be less than or equal to 500 milliseconds. This means that the next reading will be discarded if the time gap between the reading and the previous one is longer than 500 milliseconds, in order to ensure that only valid reading sequences are interpreted.

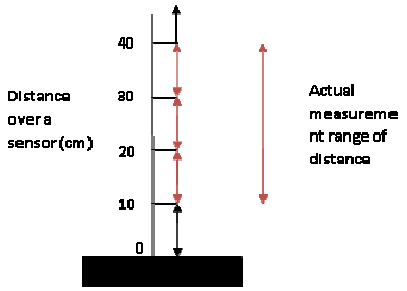


Figure 2: Actual measurement range of distance considered in our design. Red arrows are indicating the range of valid distance section and maximum distance is 40cm.

In general, we have the following condition for consecutive readings at time $t_{n+1}$ and $t_n$ in a valid reading sequence of two: $(t_{n+1} - t_n) <= Tw$ (Time window)

### *E. Reading Sequences*

The length of a reading sequence is set to a particular number for the purpose of reducing error rates. In Phidgets [15], changing the sensitivity of a sensor is used to filter the number of events that are returned to an application, by setting a minimum amount of activity before a sensor reading change event is sent to the application for data collection. This is a simple reading lag – a minimum amount of change has to occur since the last event before another event will be fired. On this basis, changing sensitivity can be used to reduce redundant sensor readings in order to eliminate the "noise" of a sequence of sensor reading values. We use this technique to adjust the speed of detectable hand movements applied to the application, by setting a minimum number of event change of sensor readings. Data rate is a refresh rate of each reading of a particular sensor, from a sensing event change to the next sensing event change. It is calculated by milliseconds and can be controlled based on application requirements. We consider varying data rates in our sensor data processing algorithms in order to maximise accuracy of gesture recognition. Research and experiments on adjusting the sensitivity and data rate of sensors have been done to find the appropriate sensitivity and data rate settings that could be best applied to our application (more results shown later). One of the results from our data rate and sensitivity experiments is that a successful gesture would take about 1 second to complete with the data rate set at 80. As a result, there will be likely twelve sensor readings for a gesture. In our design, the number of readings in a sequence is set to 10 for ease of calculation. Figure 3 represents a successful reading sequence for a given valid time window (Tw).

A *successful reading sequence*, as Figure 3 shows, contains a sequence of 10 sensor readings including sensor numbers, sensor reading values and time gaps. Time gaps for every reading in a successful reading sequence must be within the time window, by definition. However, an unsuccessful reading sequence could happen during every sensor reading. Figure 4 shows an unsuccessful reading sequence where one of the time gaps between two reading time points is longer than the time window, which is equal to 500 milliseconds. The red bar in Figure 4 represents the end of range of the valid time window from the previous reading time point $t_4$, whereas $t_5$ is out of range of the valid time window and will be considered as the end of the last reading sequence and the reading sequence will be discarded due to lack of readings (less than 10). Besides, sensors detect incoming signals at time point $t_5$, so that $t_5$ is considered as the starting point of another reading sequence. A single *sensor reading record* contains information about a successful sensor reading: sensor number which identifies the sensor, sensor value and time gap. A single *reading sequence* is a collection of 10 sensor reading records, one for each sensor reading.

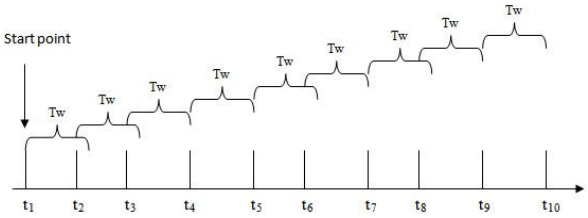

Figure 3: Successful reading sequence gvien a valid time window

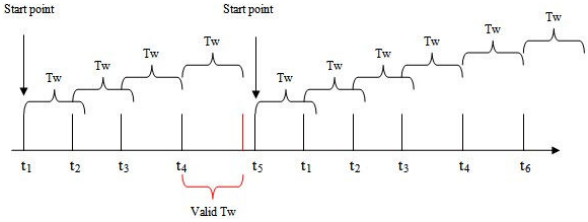

Figure 4: Unsuccessful reading sequence

## IV. IMPLEMENTATION AND EVALUATION

### *A. Gesture Detection Implementation*

When the application starts up, the application will create a sensor change event object to listen to sensor reading change events. Phidgets [15] has API in for creating such an object.

*Sensor Reading Condition and Movement Direction Check*

The range of detection distances was studied, and we considered best for the convenience of the user to have a valid distance range from 10 cm to 40 cm. After a successful reading sequence, the application will process the reading sequence to determine which movement has occurred. The application will conduct a sequential conditions check in order of horizontal movement check, static positioning check and vertical movement check. This ordering is important due to the order in which we store the reading sequence records.

Recall that the first digit of every sensor reading record contains the sensor number. The algorithm for horizontal movement check, hence, checks whether the first digits (of the records in the reading sequence) are the same, or different. Also, a *height level* was introduced into this gesture positioning and movement condition. The actual reading value from sensors may vary frequently according to the incoming analogue signal. So, we divided the valid distance ranges into segments, each segment representing a range of readings corresponding to a particular height level. As a result of that, the static positioning recognition algorithm is simplified. A horizontal movement would then mean that in the reading sequence, the sensors are different but the sensor readings all correspond to the same segment, i.e., the same height level.

After a horizontal movement check is completed and if it is found to be false, that is, no horizontal movement is recognized, the application will then proceed to a vertical height level check, in order to match and distinguish at which height level the hand movement was performed.

The third step of direction movement check is vertical movement check. A hand movement over more than one height level will be considered as a vertical movement.

*Time Window Check*

The concept of the Time Window enables a reading sequence to be viewed as recording a continuous movement. A time window check consists of two steps, which are, setting a start time and then calculating the time gap between sensor readings.

*Reading Sequences Recording and Analysis*

Reading sequences are pivotal to the application. They are recorded and analysed by the algorithms which affect the whole application performance in the application. Only successful/valid reading sequences (i.e., a succession of 10 readings satisfying the time window checks) are stored and analysed.

In the horizontal movement recognition algorithm, it will firstly detect the direction of the movement based on the first sensor number in the reading sequence. Since the sensors with lower sensor number are physically implemented to the right of the operating panel, a movement from sensors with lower sensor number to sensors with higher sensor number will be regarded as a right-to-left horizontal movement, whereas, a movement from sensors with higher sensor number to sensors with lower sensor number will be regarded as a left-to-right horizontal movement. Secondly, based on the detected directions, a sequence of comparisons of sensor numbers is recorded for later gesture recognition analysis.

By comparing sensor numbers in sensor reading records within a valid reading sequence, a mathematical algorithm is used to calculate the probability of a possible movement direction (left-to-right or right-to-left). As there are 10 records of sensor readings stored in a reading sequence, there will be 9 comparisons between those records. The purpose of the algorithm is to calculate the percentage of the number of expected results in the total number of sensor number comparisons, in order to meet the preset percentage threshold. The threshold is set to be at least 8 of 9 correct comparisons (i.e., mostly increasing/decreasing order of sensor numbers), which is 89%, for the reason that though sensor numbers tend to change smoothly, there may be several error readings which can be ignored.

In the vertical movement recognition algorithm, it will firstly record comparisons of sensor reading values in the reading sequence. An algorithm is used to compare the sensor reading values between one record and the next one from the first to the last record in the reading sequence. Here, we use a threshold of 5 out of 9, which is 56% (i.e., a majority of the sensor reading values are increasing/decreasing) for the reason that sensor values fluctuate due to sensors' reading sensitivity.

The last analysis for a reading sequence is the static position-based gesture recognition. The algorithm for static position gesture recognition was described earlier.

## B. *Other Implementation Issues*

We initially intended to use eight Phidgets distance sensors organized as a linear array panel to capture gestures. However, our experiments show that using eight sensors reduces accuracy in recognition. We show our results below with an array of 2, 3 and 4 sensors (more than 4 led to inaccuracies).

Also, Agnihotri *et al*.'s research [16] studied adult hand palm dimensions which showed that the average breadth of an adult hand palm is 84mm for males and 74 mm for females. Based on Agnihotri's research, we implemented our application with three sensors with a distance of 85 mm between two sensors to avoid detection by two sensors at the same time. Figure 5 below shows the sensors' positions in our evaluation.

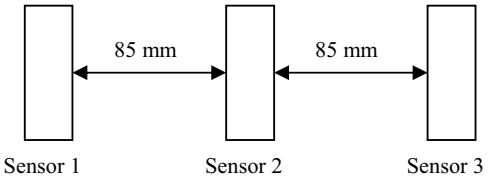


Figure 5: Position of sensors

## C. *Performance Evaluation*

Our performance evaluation focused on response times and accuracy of gesture recognition.

*Response Time Evaluation*

By measuring and recording response times over repeating gestures, the average response times for gesture recognition could be generated. Then, we compare the results with our intended design objective.

In our design, the Time Window is set to 500 milliseconds. Since a valid reading sequence contains 10 sensor readings, the maximum response time to recognize a gesture could be calculated as follows, to be roughly:

Time Window * Number of Readings = 0.5 * 10
= 5 seconds

(ignoring any calculation times which would be negligible). In fact, according to tests we performed, almost invariably, for a successful reading sequence, the average time spent between sensor readings with normal hand gestures (vertical or horizontal) is far less than the Time Window we set. The result was that the average time gap (between successive readings in a valid reading sequence) is 150 milliseconds, and so, the actual time to recognize a gesture is much less than 5 seconds: for each successful gesture detection, it takes only about 1.5 to 2 seconds to gather the (ten) sensor readings required.

*Accuracy Evaluation*

In this evaluation, we focused on accuracy of gesture recognition of the application based on our implemented recognition algorithms. We expected an accuracy of 95% gesture recognition rate in our design phase. By performing gesture recognition tests with all types of supported gestures, we recorded the result of the

recognition rates. We generated an average rate of recognition for each supported gesture.

Figure 6 below shows our results on initial experiments to determine sensor readings accuracy on various data rates and sensitivities. This enables us to choose the right data rate and sensitivity for our application.

The first two column charts in Figure 1 shows the accuracy of sensors reading in percentage based on sensitivity levels of 80 (left) and 100 (right). The horizontal bar represents the data rate of sensors. Each bundle of three columns represents the accuracy of sensors reading in percentage for two, three and four IR distance sensors arranged in order from left to right. The results show that using two sensors at a data rate of 96 provides the most accurate readings in either sensitivity of 80 or 100, with accuracy over 95%.

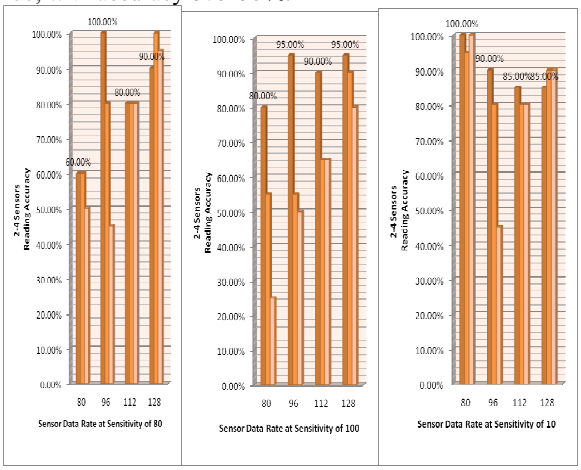


Figure 6: Sensor reading accuracy based on data rate and sensitivity

In the third chart, we also found that another set of sensitivity and data rate settings for sensors readings are appropriate for our recognition algorithm, by setting default sensitivity of 10 and data rate of 80; we get an accuracy of 100% for two sensors, 95% for three sensors and 100% for four sensors.

Our results for average recognition rate of supported gestures is as follows:

- 99% for static position-based gestures;
- 96% for vertical movement-based gestures;
- 96% for horizontal movement-based gestures;
- And 60% for combination gestures.

From those results, we could conclude that static position-based, vertical movement-based and horizontal movement-based gestures are in high accuracy of recognition over 95% and close to our target recognition rate. However, combination gestures perform far worse than the other three. Combination gestures were hard to recognize accurately as two (or more) successive valid reading sequences needed to be obtained, without a time gap between them, and it was found that it was natural to have a "rest period" between two gestures (e.g., in doing a box gesture clockwise, there is a rest period between after completing a horizontal right-to-left gesture, and the next upward vertical gesture, and between upwards and left-to-right, and between left-to-right and downwards). Our tests did not take this rest period into account. The tests were done by the first author, who developed the system.

### *D. User Inexperience*

A limited usability study was conducted by two different groups of participants based on prior knowledge of using sensors, which are categorized as known and unknown. The procedures and tasks performed for each group are the same. The known group consists of three computer science students and the unknown group consists of two business students.

Experimental results show that the known group performed well using the application. They presented high accuracy, which is over 95%, and showed quick learning of the application. Those results are close to our expectations. However, the unknown group took a bit longer to use the application. They were also unable to perform very successful interactions with the application, with accuracy below 60% in the initial stages. Mistakes in gesturing led to consecutive readings out of range of the half-second Time Window and lack of sensor readings to constitute valid reading sequences were the main problems with these participants using the application.

## V. CONCLUSION AND FUTURE WORK

We have implemented a simple gesture based recognition system for controlling applications using an array of Phidgets distance sensors (we used it to control Powerpoint slides in a university Open Day event). We note the following findings:

- Reading sequences: reading sequences are an important part of our application for the reason that the successful sensor reading records and the gesture recognition algorithms affect the application performance.
- Supported gestures: basic gesture recognition can be designed and implemented using a linear array of distance sensors, in horizontal, vertical, and static positions and combinations of movements based gestures. Horizontal gestures are from left to right and right to left. Vertical gestures are from up to down and down to up.
- Sensor sensitivity and data rate: sensor sensitivity is used to adjust the detection of speed of hand movements applied to the application, whereas, sensor data rate is to control the number of sensor readings in a particular time window. We found suitable data rates and sensitivity for normal, comfortable and natural hand gesture movements.
- Sensor reading conditions and movement direction check: our use of heuristics (without machine learning) based processing of sensor reading sequences was adequate to obtain good performance, but mainly for more experienced/practiced users.

Future work will involve improving the accuracy in recognizing gestures constituted by a combination of movements. Compositional gesture recognition will

provide a key to supporting a whole (potentially infinite) range of complex gestures made up of simpler gestures.

## REFERENCES


[1] Hewett, T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., Perlman, G., Strong, G. and Verplank, W., 2003, "ACM SIGCHI curricula for Human-Computer Interaction", Association for Computing Machinery, pp. 5, Michigan, USA.

[2] Kratz, S. and Rohs, M., 2009, "Hoverflow: expanding the design space of around-device interaction", Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services: Demos & experiences, article No. 4, Bonn, Germany.

[3] Parekh, P., 2010, "Dunhill Diary installation for world expo 2010 display", http://pinakparekh.wordpress.com/2010/07/16/our-first-gesture-interface-with-pir-sensors-and-no-camera/

[4] Ishikawa, T., Horry, Y. and Hoshino, T., 2005, "Touchless input device and gesture commands", Proceedings of international conference on Consumer Electronics, ICCE. 2005 Digest of Technical Papers, pp. 205-206, Las Vegas, USA.

[5] Benko, H., Wilson, A.D. and Balakrishnan, R., 2008, "Sphere: Multi-Touch Interactions on a Spherical Display", Proceedings of the 21st annual ACM symposium on User interface software and technology, pp. 77-86, Monterey, CA, USA.

[6] Lucero, A., Aliakseyru, D. and Martens, J.B., 2008, "Funky Wall: Presenting Mood Boards Using Gesture, Speech and Visuals", Proceedings of the working conference on Advanced visual interfaces, pp. 425-428, Napoli, Italy.

[7] Hodges, S., Izadi, S., Butler, A., Rrustemi, A. and Buxton, B., 2007, "ThinSight: Versatile Multi-touch Sensing for Thin Form-factor Displays", Proceedings of 20th annual AMC Symposium on User interface software and technology, Vol. 252, No.6, pp. 259-268, San Diego, California.

[8] Mistry, P., Maes, P. and Chang, L., 2009, "WUW – wear ur world: a wearable gestural interface", Proceedings of the 27th international conference extended abstracts on Human factors in computing systems, pp. 4111-4116, Boston, MA, USA.

[9] Wang, C., Liu, Z. and Fels, S., 2010, "Everyone Can Do Magic: An Interactive Game with Speech and Gesture Recognition", Proceedings of the 9th ICEC International Conference on Entertainment Computing, pp. 32-43, Seoul, Korea.

[10] Lee, J., 2008, "Low-Cost Multi-point Interactive Whiteboards Using the WiiMote", http://johnnylee.net/projects/wii/

[11] Schönböck, J., König, F., Kotsis, G., Gruber, D., Zaim, E. and Schmidt. A., 2008, "MirrorBoard - An Interactive Billboard", Mensch & Computer, pp. 217-226.

[12] Sharp distance sensor GP2Y0A21YK0F, 2006, "data sheet", http://document.sharpsma.com/files/gp2y0a21yk_e.pdf

[13] Acroname Robotics, 2000, "Sharp Distance Sensors Comparison", http://www.acroname.com/robotics/info/articles/sharp/sharp.html#e8

[14] Greenberg, S., 2001, "Phidgets: Easy Development of Physical Interfaces through Physical Widget", Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology, pp. 209-218, Orlando, Florida, USA.

[15] Phidgets, 2010, "Phidget Programming Manual", http://www.phidgets.com/documentation/LiveCode_API_Manual.pdf

[16] Agnihotri, A.K., Purwar B., Jeebun N., Agnihotri S., 2006, "Determination of Sex by Hand Dimensions", The Internet Journal of Forensic Science,

http://www.ispub.com/ostia/index.php?xmlFilePath=journals/ijfs/vol1n2/hand.xml