3D Gesture-Based Control System using Processing Open Source Software

Abdel-Mehsen Ahmad[1], Roba Al Majzoub[2] and Ola Charanek[3]
School of Engineering, Department of Computer and Communications Engineering
Lebanese International University (LIU)
Bekaa, Lebanon
[1]abdelmehsen.ahmad@liu.edu.lb,[2]robamajzoub@gmail.com, [3]olacharanek@gmail.com.

***Abstract*—The following paper discusses the design and implementation of a 3D gesture-based control system that can substitute any complex human-machine interface with a simpler and more familiar one. This system that can be used by any user disregarding his/her background (age, education, health, language etc.…), making it dependent on nothing more than the movements of the user's hands, thus allowing them to be free from all hardware while interfacing, and making the system surpass in its simplicity other systems that include held or touched interfaces. It utilizes the Kinect sensor's image input parts (IR projector and IR camera) for computer vision along with an open source software called Processing for reception and analysis of input data to decision making on what actions and steps to perform, and even for communicating with the database. A MySQL database will be responsible for holding the information of the system and to handle them (adding, deleting, retrieving, etc.…). Even though the system will be designed for a library to display information about the books available in it, however the proposed system design will be broad enough to allow its implementation in many other domains, making people's lives more efficient and innovative. Implementation steps are listed along with a detailed explanation of how the system works. The simulation results showing the system performance in different environment conditions and some screenshots are also displayed.**

Keywords—Open Source; Kinect; Gesture-Based Control; Processing Software; Computer Vision; MySQL.

I. INTRODUCTION

The huge leap humanity has witnessed ever since the 60s of the last century due to the super-fast development in technology has made today's public needs much wider and more complex than they ever were. However, this advancement has made the underlying systems' controls either more complex, or more expensive, or both, making them unattainable by a large number of people. This is where the need for simpler control systems and interfaces rises. Our system comes forth to heed to these needs. We propose a low-cost, gesture-based control system that depends on nothing more than simple movements of the user's hands to work, making it suitable for all people, educated, uneducated, able, disabled, rich and poor.

The designed system has computer vision implemented through the aid of the "Kinect"; "a motion sensing input device by Microsoft, which provides full-body 3D motion capture, facial recognition and voice recognition capabilities"[1]. The Kinect produces 3D images with depth information also known as a depth images. Processing software, which is an open source software and runs on various platforms, along with MySQL database system will compose the software part of the project responsible for information analysis and data manipulation.

II. RELATED WORK

Similar approaches have been adopted before seeking to achieve the goals mentioned earlier. A gesture controlled user interface called "Open Gesture" was proposed in [2] where they used a web cam with an open source program to implement their system in smart TVs for most categories of people. Even though this approach has its own advantages like affordability, and the ability of majority of people to use, especially elderly and physically disabled people, however it has certain drawbacks. First of all the visual input device is a webcam, meaning that for visual analysis, image processing will be used for detection of gestures, which is computationally expensive and not as effective as 3D scanning methods. The webcam like any other 2D camera can only work properly in well-lit places, where dim-lit and less bright places present challenges for the "vision" of the system and for the analysis of the input information, and that is something we addressed with the use of the Kinect (which uses infra-red images thus can work in dim lit and even dark places). Yet another drawback is the fact that the designers of the system assume that all the elderly and the users of the system are educated, or have a certain level of education and that is not always the case. Even though the system we propose in this paper is for a library, however the gesture control can be performed over anything and almost everything we want with some small changes.

Another approach was presented in [3], which uses the WiiMote from Nintendo to control smart homes (appliances, lights, windows…), where the WiiMote's 3D accelerometer was used to detect the gestures and along with it a classification framework composed of machine-learning algorithms like Neural Networks and Support Vector Machine (SVM) was used for gesture recognition classification and training. This approach has a high level of accuracy for

978-1-5090-4580-8/16/$31.00 ©2016 IEEE

gesture recognition and classification due to the use of the classification framework, however a setback lies in the use of the WiiMote, where it restricts the gesture recognition for the user to only the time when the controller is held by hand, and which after long periods may result in inconveniences for the user.

Yet another approach for gesture control was presented in [4]. This approach uses MEMS, Micro Electro-Mechanical Accelerometer Sensors to move wheelchairs of elderly and physically disabled people. The accelerometers are attached to the fingertips and back of the hand, where these detectors are very sensitive to tilts, thus the user can simply use finger and simple hand gestures to allow the user to move his/her wheelchair without any aid required. This approach is a very good one which greatly simplifies the control of wheelchairs especially for elderly and people with disabilities; however it limits this control to wearable devices, and limits the scope of the system to only specific uses.

III. Proposed System

A. Designed Specifications

Since the system design targets the majority of people, we had to take into consideration the fact that users will vary between old, and young, educated and uneducated, able or with disabilities, so the controlling process had to be performable and known to all users with ease without resorting to difficult physical actions. Natural User Interface (NUI)[1] was the answer.

From the many choices of NUI we chose to utilize hand gestures since they are familiar and universal to all people and since even small children learn to use gestures before they even learn to talk.

For gestures to be used for control, we had to implement human vision into the computer using an input sensor, and here another requirement surfaced, which is the affordability of the system by people. Since most systems with good interfacing properties are expensive, our goal was to create a system with commendable price and high interactivity, all which was provided by the Kinect sensor.

Another advantage of the Kinect sensor was its ability to catch great depth images in dim lit or unlit places, which is a property not available in any of the other image capturing sensors.

As for the programming language, we used Processing, which is a free open source programming language and runs on Mac, Windows, and GNU/Linux platforms. Processing has many libraries related to Kinect, and because it was easy to learn and handle object oriented programming language, it allowed us to have more flexibility when dealing with Kinect input and manipulating information and output.

The following figure (Fig.1) represents the interaction between hardware components of the system designed:

[1] *a system for human-computer interaction that the user operates through intuitive actions related to natural, everyday human behavior.*

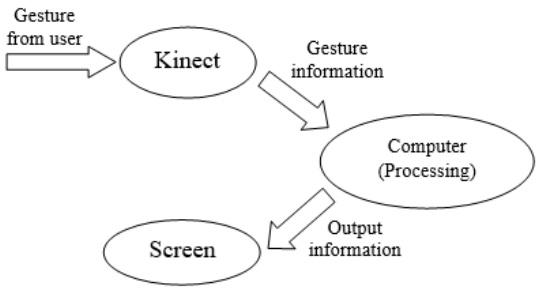

Fig. 1. Hardware components of the system

IV. Implementation

Because our goal was to create a system controlled by gestures we needed a gesture recognition algorithm to start with. So we decided to use a pre-made version that was available online to save time and effort and to focus on the main idea of the project.

This resulted in two approaches to perform control over the system, one was 2 dimensional (2D) and the other was 3dimensional (3D) based on the algorithm used for the detection of gestures.

A. 2D Approach

The first library we tried to use for gesture detection implementation in Processing was the FingerTracker library which is a Kinect-related library that allows the detection of fingertips through slicing the space detected by the Kinect into layers of 2D surfaces and then determining the fingertips through scanning each 2D surface and finding "inflections in the curvature of the contour" [5] then drawing red ellipses in their positions, as can be seen in Fig. 2.

Fig. 2. Fingers tracked using the FingerTracker Library

Now since we can detect fingers, we decided to use finger gestures and movements to perform the control in the system.

Our first step was to eliminate the depth image (the grey-scaled 3D-image captured by the Kinect seen in Fig.2) from the displayed frame by making the program redraw the contour and fingertips with every frame obtained from the Kinect, which is almost 30 fps (frames per second) on a black background, and thus allowing us to observe real-time fingertips tracking of our fingers and the contour of our hands. Then in the positions of the fingertips we inserted special 2D graphics known as "Sprites" as displayed in Fig.3. These Sprites are 2D images that have special properties with

978-1-5090-4580-8/16/$31.00 ©2016 IEEE

boundaries and libraries that allow us to detect their collision with each other, set their properties, motion, speeds and so much more. The library used here is called Sprites library, and allows us to control all the previously mentioned properties of sprites.

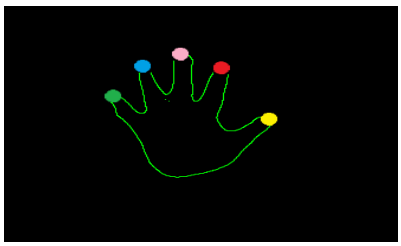

Fig. 3. Sprites placed on the fingertips

Next, images of books were inserted as sprites as well (see Fig.4) and an algorithm was created for collision detection between them and the fingertips sprites so that certain actions can be executed upon detection.

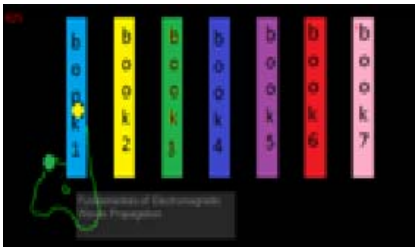

Fig. 4. Book information displayed upon collision detection between the sprite of a fingertip and that of a book

A loop was used to iterate over the seven pictures given, then a condition checks for any collision between any book and a predefined fingertip –which is shown using the yellow sprite in Fig. 3. If the condition is satisfied, the name of the book that the finger sprite collided with will be printed on screen (see Fig.4). Then the loop iterates over the books again to check for collisions. The following algorithm describes the process:

Step 1: start the loop
Step 2: Check for collision between sprite boundaries.
Step 3: If condition is true, then perform a certain action.
(Here display information about the book in a text area)
If false, do nothing.
Step 4: Check the boundaries of the next book sprite.
Step 5: When loop is finished, start the loop again.

Other similar methods were developed to check for collisions between the books and more than one fingertip, but this time with another condition; the distance between the boundaries of two consecutive fingertip sprites (to represent the pinch gesture) which will result in displaying information previously saved in a String array about the book, the following algorithm describes the mode of action of the loop:

Step 1: Start the loop
Step 2: If Collision between first fingertip and book i is true, then check if collision between second fingertip and same book boundary is true.
Step 3: If both conditions are satisfied, then check if distance between fingertip 1 and fingertip 2 is less than a certain threshold.
Step 4: If true then display an output. (Here the output will be an open book that contains all the information about the book i)
Step 5: If any of the above conditions is false restart the algorithm till all conditions are satisfied.

Figure 5 displays how the software components in this approach work with each other.

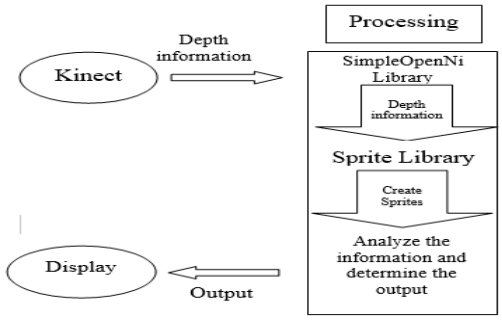

Fig. 5. Software Components of the 2D approach

Another option that was available for the implementation of our project was within the SimpleOpenNi library which is the "Hands" sample code. This code allowed us to detect three hand gestures: Wave, Click, and Hand Raise. Upon the detection of any of these three gestures, the detected hand is given an ID number between 1 and 7 and a coloured point in the middle of the upper part of the hand palm is displayed on the screen where each hand ID had a unique colour, as can be seen in Fig. 6.

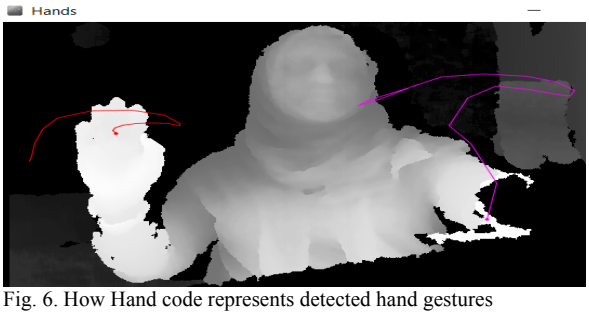

Fig. 6. How Hand code represents detected hand gestures

This was a great help for us to develop our program. We then found out that each gesture is assigned an integer number from 1 to 3, where Wave = 0, Click = 1, and Hand_Riase = 2. For the control we wanted to perform we only needed two out of the three gestures, so the click gesture was used for selection and the wave gesture was used for going back one step.

Since gesture detection was available, our next step was to allow the system to respond to a certain gesture in a certain position. The project is developed for a library, so gestures

978-1-5090-4580-8/16/$31.00 ©2016 IEEE

were needed for the selection of topics, books, and for navigating forward and backward between topics and books.

A small description of the system along with some screenshots will be displayed for better understanding of how the system works then we will go further into the details of the program and database implementation.

The user faces a Kinect sensor and a Screen that displays his depth image. With a wave of his hand a "START" button will be displayed in the middle of the screen. (Fig. 7)

Fig. 7. After the wave of the hand, the "Start" button appears

The following algorithm applies to all the next steps where it checks to see if certain conditions are satisfied then it performs a certain action. The conditions used were Boolean variables, certain positions and types of gestures, where if a certain gesture is performed in a certain position then a specific output will be displayed, whereas if the same gesture is performed elsewhere, then a different output will be displayed, the algorithm is as follows:

Step 1: Start conditional loop

Step 2: Check if condition 1 is true

Step 3: If true then check condition 2 (gesture type) is true, otherwise exit loop and recheck condition 1 till it becomes true.

Step 4: If condition 2 is true, then perform a certain action (we can print output / display images / display information about a certain book….)

Step 5: If false then keep on checking till both conditions are true.

Where the gesture type defines the gesture recognized: wave, or click.

As the user clicks the start button, a new screen will appear with the topics of the available books displayed on the left side of the screen, and empty shelves in the middle of the screen, as seen in Fig.8. Here the condition was to perform the click gesture within the boundary range of the start button.

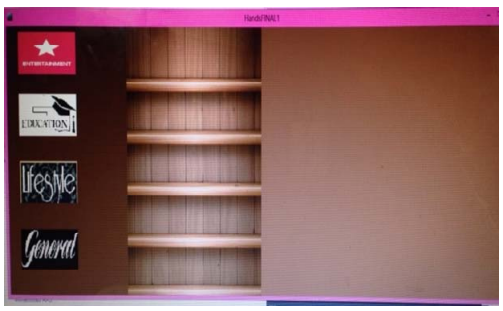

Fig. 8. Screen displayed after the click on the start button

Next the user can select any of the displayed topics he/she is interested in through a click gesture on the required topic.

As a response to this gesture pictures of the available books concerning this topic will be displayed on the empty shelves as seen in Fig. 9.

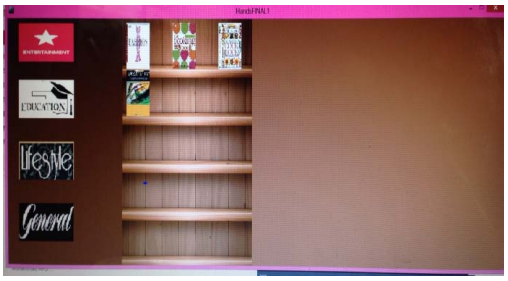

Fig. 9. Books displayed on the shelf

To check the information of one of the displayed books a simple click on the book displays a larger image of the book along with the information (author, ISBN, summary, position within the library…) on the right hand side of the screen as seen in Fig. 10.

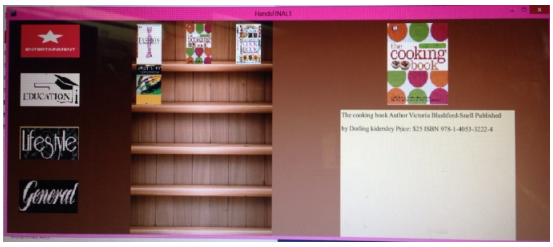

Fig. 10. Information displayed about the selected book

Going back to the previous menu can be done with a simple wave of the hand. The same gesture can take us back one step wherever we are within the program, as seen in Fig.11.

Fig. 11. Going back to previous menu

Note that all the above mentioned steps can be performed using the second algorithm mentioned previously with simple modifications to the conditions.

Figure 12 displays the software components of the 3D approach of the system.

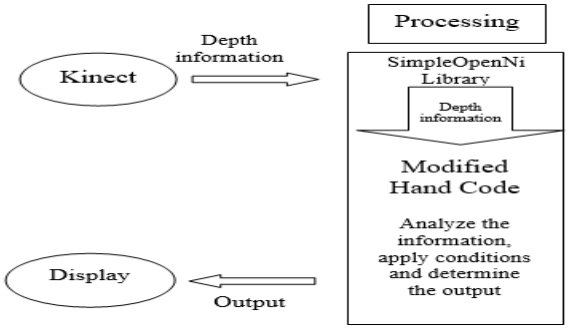

Fig. 12. Software components of the 3D approach

978-1-5090-4580-8/16/$31.00 ©2016 IEEE

As mentioned earlier, MySQL database was created to store the attributes and information of the books to be displayed on screen. All the pictures were placed in a folder and the other information were to be retrieved from the database. MySQL server 5.0 was used for information storage.

The created database had two tables within it: The first table contained the IDs and names of the pictures corresponding to the available categories which are “education”, “entertainment, “lifestyle” and “general” as seen in Table I.

TABLE I. "TYPE" TABLE IN "LIBRARY" DATABASE

ID	Name
1	Entertainment
2	Education
3	Life Style
4	General

The second table had the following attributes: id INT, typecategory VARCHAR, name VARCHAR, info VARCHAR, and bigpic VARCHAR. (See Table II)

TABLE II. "BOOK" TABLE IN "LIBRARY" DATABASE

id (int)	typecategory (string)	name (string)	info (String)	bigpic (String)
1	Entertainment	Fashion.jpg		BigFashion.jpg
2	Entertainment	cooking book.jpg		Big cooking book.jpg
3	Education	english.jpg		Bigenglish.jpg

The attributes "id", "typecategory", "name", "bigpic” and "info" correspond to the identity of each book, category / topic of the book (entertainment, education …), path of each of the small sized pictures of the books that appear on the shelves, path of the large sized pictures of the books that will be displayed on the right hand side of the screen when the user clicks on the small image of the book and the description of the book (title, authors, ISBN...). After filling the tables with the required information, we should be able to access these information from within the Processing, so a function was created for retrieving the data which returned the desired information in a string array declared as a global variable (so that it can be accessed from any function).

The first action performed in this function was connecting the Processing to MYSQL server, then checking whether this connection is set, through using a conditional if where the statement returns true if the connection process is successful, and the condition signals the execution of a loop that performs the following actions:

- Performing a query: Selecting all rows from the table specified in one of the attributes.
- Iterating over all of the selected rows by checking each time whether there is a next entry, and that the table has not reached an end.
- Getting the "attribute" value of each entry, storing it in a string, and then mapping it to a global string array, which will eventually store all retrieved values.
- Incrementing the loop to hop to the next position in the array, and when the iteration is done returning the array "retrieved".

This "retrieve" function was used within Processing to retrieve each of the:

- Paths of the categories’ pictures
- Books’ Description information
- Paths of the small sized pictures
- Paths of the big sized pictures

Note that as we change the conditions within the retrieve functions we can retrieve the information we want from the database.

This type of implementation has therefore made this control system quite broad where if we wanted to change the application field of the system from a library management system to another different domain, then the users must only supply the corresponding database containing the paths of the different pictures of the items they want to display, in addition to the folder containing these pictures to be placed in a certain directory from which the images will be loaded from, or even upload different contents within the database to correspond with the action required to be performed, thus extending the range of the system’s application field.

V. SIMULATION RESULTS

A. Test Cases and Acceptance Criteria

To consider the system as acceptable for interfacing it should satisfy the following criteria:

- Be able to detect gestures without too much effort from the user, and without any delays.
- Be well organized so that the user wouldn’t be confused about the information or about how to select items.
- Use gestures that are easy and familiar to people so that all users would be able to perform them without effort.
- Limit the range of gesture performance such that it wouldn’t be too wide, so that the hand movements wouldn’t strain the user, and they must be swift and not involve too much movement.
- Have a high percentage of accuracy and fidelity

The tests were performed on both approaches mentioned before, the 2D and 3D approaches.

B. Simulation Results

As for the 2D approach, even though it involves the finger gestures which are quite common among most people, still the continuous flickering of the sprites assigned to the fingertips and the non-consistency in the detected IDs of the fingers made collision detection much harder and reduced the accuracy of the system significantly, thus removing this option from the list of valid approaches.

978-1-5090-4580-8/16/$31.00 ©2016 IEEE

As for the 3D approach, the tests performed for the response time, gesture detection and usage of the proposed gestures have proved the system to be well compatible with the criteria mentioned above. The range of gesture performance has also been detected to be good and reachable by all the users who have performed this test which included users of different ages and different educational levels. The gestures used, namely the waving and clicking were very well detected and responded to, on condition that the user faces the Kinect head on without any tilting and without any obstacles. The accuracy has increased significantly from the first prototype, and the fidelity problem we first faced was solved by ensuring that the system can be used over and over with the same good outcome.

The figures previously displayed were screenshots captured during the simulation of both approaches, where each of them displays exactly how the system responds to the actions of the user.

The following graph displays the simulation result for response of the system to users of different ages and educational backgrounds, where the users were classified into three age categories: less than 10 years, between 10 and 40 and older than 40 (see Fig. 13). The vertical axis represents the number of trials of each user when performing the gesture for the first time and the vertical one displays the age groups.

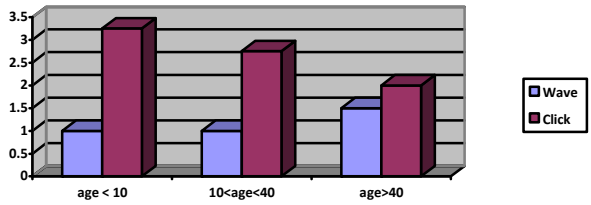

Fig. 13. Graph displaying number of trials for each gesture with respect to age

We tried the System without any lights, and the results were great, and even in well-lit places, the system still functions properly, however, the only set back is when the environment is lit by the sunlight, which has too much IR radiation, making it difficult for the Kinect to "see" its surroundings well, and so making gesture detection less feasible.

Fig. 14 displays the response of the system in different lighting conditions and on certain distances from the user. As we can see in the figure, the response of the system is optimal at 2.5 m and it performs properly at that approximate distance in well-lit places, whether it's lit by sunlight or by electrical lights, however, outdoors performance decreases due to the high level of the IR radiations.

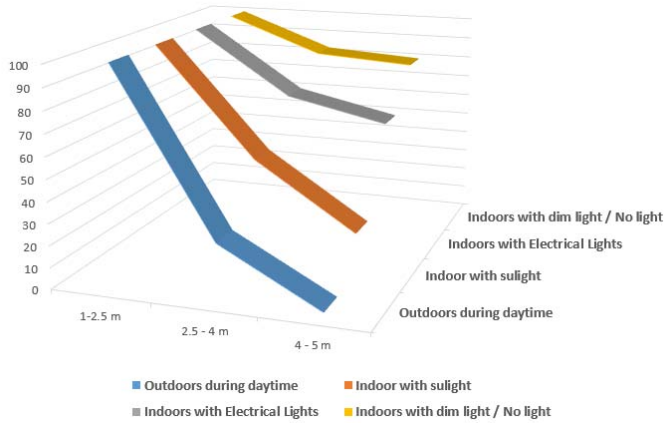

Fig. 14. System Response in different lighting conditions and different distances

Another problem we faced in this implementation which was the calibration of the sensor distance from the user, where any change in the distance between the user and the sensor had greatly affected the system. This can be resolved by keeping the sensor stationary.

The accuracy of the system was very good, however for better results new algorithms can be created specifically for the purpose of this system, which would enhance the accuracy and hence the overall system response.

VI. Conclusion

The outcome of this project were satisfactory for a low cost, easy interface control system prototype with good accuracy and response which can be implemented anywhere to control systems easier, however it can be improved through further studies and more specialized hand detection algorithms.

References

[1] Kinect. [Online] Wikipedia. https://en.wikipedia.org/wiki/Kinect

[2] Bhuiyan M. and Picking R, "A Gesture Controlled User Interface for Inclusive Design and Evaluative Study of Its Usability", Journal of Software Engineering and Applications, 2011, 4, 513-521.

[3] Neßelrath R., Lu C., Schulz C. H., Frey J., and Alexandersson J., "A Gesture Based System for Context-Sensitive Interaction with Smart Homes", Ambient Assisted Living, Springer, 2011, pp 209-219.

[4] Vishal V. P., Nikita S.U., Darshana P. M., Nikita R. I., and Pragati P. M., "Hand Gesture Based Wheelchair Movement Control for Disabled Person Using MEMS", International Journal of Engineering Research and Applications, 2248-9622, Vol. 4, Issue 4(Version 4), April 2014, pp.152-158.

[5] Borenstein G. [Online] "FingerTracker", http://makematics.com/code/FingerTracker/.

978-1-5090-4580-8/16/$31.00 ©2016 IEEE