TECHNOLOGICAL EDUCATIONAL INSTITUTE OF CRETE

ENGINEERING SCHOOL

MSc IN ADVANCED PRODUCTION SYSTEMS AUTOMATION AND ROBOTICS

MASTER THESIS

Gesture Based Human-Computer Interaction Using Kinect

VARDAKIS EVANGELOS

R.N.: MTH55

1.

Supervisor
Dr. Makris Alexandros

HERAKLION, CRETE, GREECE
MARCH 2017

Acknowledgement

This project started in February 2016 and finished in March 2017. This period includes 2 months of research, 8 months of building the application, 1 month of experiments and 2 months of writing this thesis.

I would like to thank my supervisor professor Dr. Makris Alexandros for his guidance and help in the implementation of this project during all this time. My professor Dr. Drakakis Emmanuel for granting me with a place inside the institute to work and providing me with a personal computer of the needed specifications. My professors Dr. Fasoulas Ioannis and Dr. Kavvousanos Emmanuel for providing me with the kinect sensor. Finally, I would like to especially thank the 23 people who volunteered to join the experiments of this project.

Abstract

This thesis introduces the design and the implementation of a gesture based human-computer interaction system which gives to the user the ability to do basic computer operations without input devices such as computer mouse or keyboard, but by performing gestures with his hand in the air.

The proposed system is based on the FORTH Hand Tracker (FHT) which models the human hand and provides real-time information about its pose in 3D space. To replicate the mouse operation the proposed system uses as input the hand's position taken by the FHT and transforms it to a coordinate system aligned with the computer screen. This way the user can reposition the cursor by moving his hand in front of the desirable position on the screen. Furthermore the system recognizes several gestures that are assigned to specific commands that are typically given by standard input devices (i.e. mouse, keyboard). Gesture recognition is based on features extracted from the FHT pose and concern distances between individual hand parts, such as fingertips, in the 3D space. The temporal evolution of the recognized gestures is modeled using Hidden Markov Models (HMM). Each gesture corresponds to a different HMM while the optimal HMM/gesture is calculated using the Viterbi Algorithm. The gestures that the system recognizes correspond to the following commands: left-right-double click, drag and drop, zoom in and out, volume up and down.

Experiments with data taken by a group of users in real conditions are showing the efficiency of the proposed method, with 90% success in the total sample population.

Table of Contents

Table of Pictures

Table of Lists

Section

1

Introduction

1.1 Science of Computer Vision

Computer Vision is the scientific field that concerns the extraction of semantic information from the raw images provided by optical sensors, much like the processes the human visual system uses to transfer and translate optical data using the human brain (1). The study of Computer Vision first begun in the late 60s as part of the research for artificial intelligence systems (AI). Computer Vision can be basically used as an automated input method for computers (much like a keyboard or a mouse) with major application in various fields of study such as automation, robotics, medical technology as well as entertainment field.

1.1.1 Technical Information

Thru Computer Vision a computer processes digital information, such as digital images or videos (both of which are technically the same since a video is nothing more than a series of images), captured by a digital camera. Digital information is nothing more than a sequence of 1's and 0's (binary information) therefore computers process a digital image as pure numerical information which in turn they output in a form that can be comprehended by the human user thru the use of a graphical user interface (GUI). A digital grayscale image of dimensions 640x640 pixels is a two-dimensional matrix of 640 rows and 640 columns which in total gives us 409600 cells. Each cell contains different numerical data which corresponds to a different shade of gray. A color image is produced using the same principle but there are 3 two-dimensional matrixes of the same size, where each matrix corresponds to a different color, one for the shades of red, one for the shades of green and one for the blue color (RGB image). Combining these three colors allows us to reproduce the entire color palette.

The concept of Computer Vision requires the conversion of information regarding the color and shape of an object to numerical data that can be processed by a computer. Something as basic to humans as the color of an object must first be converted into numerical information so that Computers can process this information and in turn translate it into the corresponding color. The shape of an object is derived by the patterns

of its color scheme. For example, a square is nothing more than the collection of neighboring pixels that have the same color information and follow the pattern of a square shape. A computer basically translates these patterns and gains the extra information regarding the shape of the object. If the distance of an object (shape) from the camera is known, information regarding the scale can also be derived using similar principles. Modern methods of Computer Vision can provide the information of the distance from the camera. Some of them are the laser distance measurement, the stereoscopic camera and the method used in this project, the structured light depth sensor [Kinect Sensor]. Knowing the depth of an object we can find out its three dimensional shape, its size and its position in 3D space.

By using databases of different objects, shapes and colors, a computer recognizes specific items. Even if the items vary in shapes we can use specific features to recognize the objects. For example not all the faces have exactly the same shape, but if we have two items-features looking like eyes, two looking like ears and one which looks like a mouth and if all these features are placed in a specific manner we can conclude that the item corresponds to a face.

1.1.2 Computer Vision Applications

Some of the Computer Vision applications include the automatic inspection of production lines in the Industrial and manufacturing fields and controlling processes in industry automations and robotic systems. In addition computer vision can be used in assisting humans in identification tasks, such as medical examinations and detecting events, such as counting the number of people that have passed in front of a camera. Computer vision can also be used in modeling objects or environments, such as mapping and in the navigation of autonomous vehicles or robots. Finally, Computer Vision can be applied to directly interact with a computer which is the area of study of this dissertation.

1.1.3 Tasks and Processing Methods

All these aforementioned applications require one or more Computer vision tasks. The term task refers to the various functions that must be performed in order for the information to be translated for a computer and in turn be used to initialize the desired response. In more simplified terms the computer must turn the input information into a desired output effect. One of these tasks is “Recognition”, such as object recognition, handwritten digits recognition, face recognition in photo shooting and hand poses recognition in the case of our study. Another task is “Motion Analysis” which is processing a sequence of images to estimate the velocity and moving direction of an object, such as in tracking applications. Other tasks are “Scene Reconstruction”, used in panoramic photo building and modeling applications and “Image Restoration” which is

used to remove noise or motion blur from images, or even to fill missing parts of an image.

In order to perform a Task, information must be first acquired from the surroundings and then processed in order to produce the desired outcome. A typical processing method in computer Vision systems is as follows. Firstly we have the "Image Acquisition" which refers to digital information captured by cameras and depth sensor systems which can be interpreted as 2D or 3D and grayscale or RGB images, depending on the hardware used and the requirements of a specific application. Secondly, we have the "Pre-Processing" of the data which refers to the simplification of the information gathered thru "Image Acquisition" in order to maximize the efficiency of the task by discarding unnecessary information as well as adding additional information in corrupted or missing places. For example noise reduction in an image or adding coordinate information to a transmission. Thirdly, we have the process of "Feature Extraction" which refers to the recognition of distinct patterns of information such as lines, edges, corners, blobs and points. After having identified the general information of an object we use the process of "Detection/Segmentation" with which we can isolate the desired piece of information and discard the unnecessary bits. Next we have the "High-level Processing" phase in which all the gathered information is combined to classify and detect specific objects and extract targeted information from it, such as pose or velocity. The last step is the "Decision Making" which refers to the way the information is used by the system and it varies depending on the use and purpose of the application. In the case of a gesture based human-computer interaction system the "Decision Making" phase refers to whether or not the computer can recognize a specific gesture and initialize the desired command that is assigned to the gesture.

1.2 Thesis Goal

This thesis goal is the creation of a Computer Vision application allowing Human-Computer Interaction (HCI) through hand gesture recognition. The user is able to operate basic computer functions without the use of touch method devices, such as mouse, keyboard or touch screen, but by performing gestures in front of a camera.

The proposed approach has several applications: (i) Computer tele-operation. In a working environment that the employees need freedom of movement it would be useful to operate a computer from anywhere in a room. Even in our homes, such an application would make life easier by giving us the ability to operate a computer without being forced to sit in front of a keyboard. (ii) The restriction of input units and therefore the size of computers would be reduced considerably. Especially in a company or an industrial unit that many people use computers, each terminal could have a very small size sensor

as input and a small size projector as the output device. (iii) In virtual reality applications the user will be able to immerse much more in the virtual world since all the commands could be input without the use of touch method devices further emulating the way we move through a space in real life. (iv) In robotics , new ultra-sensitive machines can be created that will be able to simulate with high accuracy the movement of various human parts, for example a doctor could operate on a patient with a contagious disease by using robotic arms from a safe distance . (v) For hygiene reasons in the usage of public computers such as ATMs and vending machines.

The goal is to add the ability of gesture recognition to the preexisting "FORTH Hand Tracker" application [FORTH Hand Tracker] which was developed in Federation of Research and Technology – Hellas (FORTH). The application so far has the ability to track a human hand in 3D space and to generate visual images of its position and pose using a 3D hand model.

The developed "Hand Gesture Recognition" application's basic function will be the emulation of mouse controls with hand gestures, since thru the use of a mouse most of the functions of a personal computer can be performed. The movement of the hand simulates the movement of the mouse while the various mouse buttons will be simulated by specific gestures while adding some gestures that correspond to commands that cannot be performed by a mouse (such as zooming in and out). We will even be able to type using a virtual keyboard on screen giving as even more control over the more intricate functions of a computer. Of course not all possible commands have been emulated since the aim is not to create a fully commercial application but a prototype that can be used as a basis in future research. The commands chosen include some very basic ones such as the left, the right and the double click mouse button commands, the drag and drop commands, zoom in and out commands and volume up and down commands. Extra consideration was given to the gestures chosen as command inputs. To make the application more practical and user-friendly simple hand gestures are used that can be performed by everyone despite the shape or size of their hands.

1.3 Related Work

Several papers have been published concerning applications that use Computer Vision for Human Computer Interaction most of them suggesting that hand gestures are the ideal input method.

In the published paper (2) an RGB camera is used to recognizing gestures in a single 2D layer. Each gesture/input method corresponds to a certain number, for example the closed fist gesture is represented by 0 (zero). By corresponding a different number a

specific hand gesture complex commands can be given by sequencing different hand gestures one after the other. The application works with numerated choice menus which are activated by performing the corresponding gesture of the appropriate number. The computer functions are organized in a form resembling tree branches where, what we basically have is a series of main menus each leading to series of sub-menus and so on. The main menu has generic choices where you can choose the category of function you want to use (for example one of the options corresponds to mouse functions while a different one corresponds to the system functions of a computer). This method can give us full control of all the operations of a computer since every command can be inserted by choosing the appropriate option which can be found in one of the sub-menus.

In the published paper (3) the computer cursor follows the movement of the index fingertip while the other functions are activated with the thumb. The left click command is activated by opening the thumb, the double click command by opening the thumb rapidly twice and finally the right click command is performed by keeping the thumb in the open position for a few seconds. So the application works by identifying changes made to the main object, which in this case is a closed fist, and the timing in which these changes occur.

In the published paper (4) we have a model based hand tracker without depth information. In order to fully map the correct shape and movement of the hand an accelerometer is adapted on the hand itself which gives data about the positioning and rotation of the hand. Once the hand has been calibrated it uses the 2D data from the camera and the accelerometer data to figure out the hands position in 3D space.

In the case study (5) a Microsoft kinect sensor is used to achieve human/computer interaction by identifying full body placement and movement in 3D space. They have not built their own model but they use the one built for the Kinect for human body motion recognition. They use the information to activate computer functions limited to slideshow presentations. The functions are limited to cursor movement according to the hands position and scrolling through the pages of a slideshow by using the other hand. The Kinect's human body motion software is also used in (6) published paper where it is used to fully control the application "Google Earth" by Google. Functions such as moving forward, backward, left and right on the 3D model of the earth are achieved by hand movement and Zooming In and Out is done by moving both hands together and apart respectively. Finally in "street view mode" the user can rotate the field of view of the observer by performing the gesture of turning a car wheel.

The final published paper (7) is that of for the Foundation for Research and Technology - Hellas (FORTH) in which two different systems for Human Computer Interaction are used simultaneously. The first is a 2D hand gestures recognition system using a single camera and the second is a 3D system using stereoscopic cameras. The hand is not recognized as a single object but as a series of connected blobs which are distinguished from the surroundings by isolating the areas exhibiting skin colors properties. The open fingers are basically nodes that can be detected by any distance since the shape is evaluated at several scales. Much like the previous studies examined the number of open fingers corresponds to the activation of different computer functions. In a 2D system both hands are required for the operation of a computer where one hand corresponds to the cursor movement and is named pointer hand while the other hand corresponds to the rest of the functions. Obviously both hands can be used for both functions depending on the users preferences where specific functions are always assigned to the "pointer hand" and different function are assigned to the second hand. When all five fingers of the pointer hand are open the right click command is activated while the same gesture performed by the second hand activates the hold down left click and by closing any number of fingers the left click command is simulated. The double click command is activated by holding three open fingers using the second hand. In a 3D system only one hand is used for all the functions and movement of the cursor. To activate the left click command you have to close all the fingers and move your hand closer to camera. When you move it back to the reference point the left click command is released simulating a simple left click of a mouse. Similar to the left click command, the right click command is activated with the same motion but with all five fingers in the open position and the double click command with three fingers in the open position.

1.4 Implementation Process in Brief

In our case we use the FORTH's model based hand tracker [FORTH Hand Tracker] to track the hand's position and pose in 3D space using a full 26 degrees of freedom (DOF) hand model. The input method of optical data is Microsoft's Kinect RGB-D sensor [Kinect chapter] which is capable of 3D recognition. Using the parameters of the hand model we track the position of the palm and each fingertip. The cursor follows the palm's position and movement. The distances between the palm and each fingertip or between fingertips are used to recognize the gestures [poses and gestures]. Unlike (2) we do not rely in the number of fingers closed or exposed but in the use of specific fingers in a specific manner. For example the Left Click command is activated when specifically the index fingertip closes in to the palm. In addition, the gestures used are not static poses that correspond to a specific command but the transition between poses must be detected for the command to be activated giving the user a more organic and natural feel to the whole process while making the commands ever more controlled since wrong inputs due to human error are reduced. We use Hidden Markov Models (HMM) [HMM] to model the gestures through the transitions between all the intermediate poses performed to

reach each specific gesture. Additionally a specific algorithm, called the Viterbi Algorithm [Viterbi Algorithm], is used to determine which gesture is more likely to be performed depending on the intermediate poses detected raising the accuracy of the application even further while minimizing the possibility of inputting a wrong command. Once a defined gesture is detected, the appropriate function is activated by using the preexisting open source software called "win32api" which is a library for virtual mouse and keyboard commands. The programming language used is the "Python" language in coalition with the "Open CV" library which is used for image processing. The input method for data acquisition is the Kinect Sensor and we use the "Prime Sense's Kinect SDK" software to connect the device with a personal computer (this is required because the device is designed for a video game console and not for PC applications) and the OpenNI is the software used for image capturing.

Section

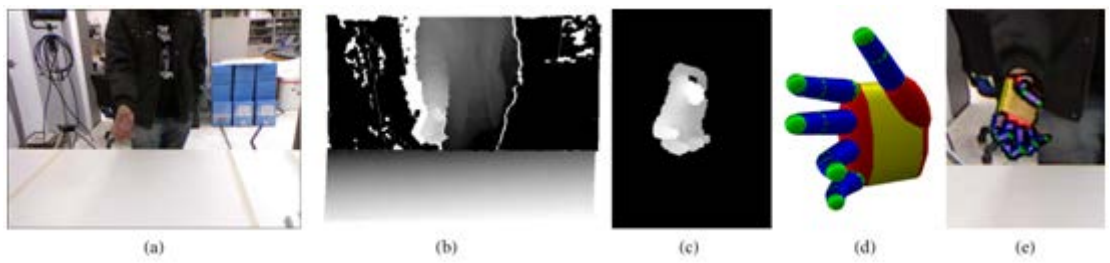
2

Background

2.1 FORTH Hand Tracker

The Hand Tracker is an application developed in the Computational Vision and Robotics lab., Institute of Computer Science, Foundation of Research and Technology – Hellas (FORTH). It is solving the problem of tracking the 3D position, orientation and full articulation of a human hand (8) (9). It is tracking the hand without the need of glove or other marks of observation.

It is a model-based hand tracker that provides a continuum of solutions and it is not limited to a discrete number of hand poses, but it can track the hand at any position and rotation of its individual parts. Particle Swarm Optimization (PSO) (10), hypothesize and testing approach is used. The objective function is based on rendering and comparison with RGB-D observations. Microsoft kinect [kinect Sensor] is used to provide depth information.

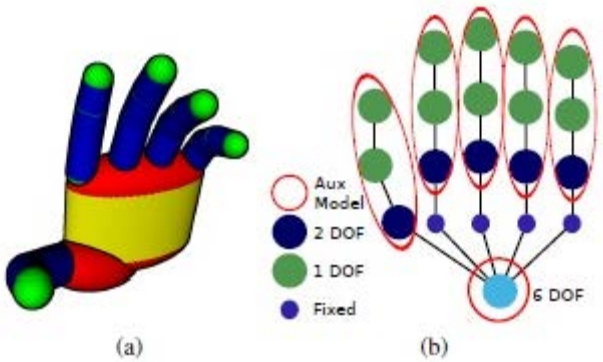


Picture 1. Graphical illustration of the proposed method. A Kinect RGB image (a) and the corresponding depth map (b). The hand is segmented (c) by jointly considering skin color and depth. The proposed method fits the employed hand model (d) to this observation recovering the hand articulation (e)

2.1.1 Hand Model

The hand pose is described by a 27 parameters vector that encodes the 26 degrees of freedom (DOF) of a human hand. The three first parameters are the X, Y and Z position

of the hand palm in 3D space. The four next parameters are the quaternions which represent the palms rotation. The other 20 parameters are allocated to 4 for each finger, and represent its rotation in three points, while the translations between them are prefixed for the hand model. The hand model is built with cylinders for the fingers, spheres for the palm and the fingertips and joints between them. Each joint represent a rotation point and each cylinder the translation between joints.



Picture 2. The employed 3D hand model: (a) hand geometry, (b) hand kinematics

Hand articulation tracking is implementing by estimating the 27 hand model parameters that minimize the discrepancy between hand hypotheses and the actual observations. To quantify this discrepancy graphics rendering techniques are employed to produce comparable skin and depth maps for a given hand pose hypothesis. An objective function is formulated and a variant of Particle Swarm Optimization (PSO) is employed to search for the optimal hand configuration, which gives the output parameters for each frame.

2.1.2 Evaluating a Hand Hypothesis / Objective Function

Having a parametric 3D hand model are estimated the model parameters that are most compatible to the visual observations. Given a hand pose hypothesis h and a camera calibration information C, a depth map $r_d(h,C)$ is generated by means of rendering. By comparing this map with the respective observation o_d is produced a "matched depths" binary map $r_m(h,C)$. This map is compared to the observation o_s, so that skin colored pixels that have incompatible depth observations do not positively contribute to the total score. There are incompatible depth observations when foreground model pixels that are background in the observations and vice versa.

The discrepancy between the observed skin and depth O is calculated

$$E(h,O) = D(O,h,C) + \lambda_k K_c(h)$$

Where λ_k is a normalization factor and k_c adds a penalty to kinematically implausible hand configurations. The function D is defined as

$$D(O,h,C) = \frac{\sum \min\left(|o_d - r_d|, d_M\right)}{\sum(o_s \cup r_m) + \varepsilon} + \lambda\left(1 - \frac{2\sum(o_s \cap r_m)}{\sum(o_s \cup r_m) + \sum(o_s \cap r_m)}\right)$$

Where the first term penalizes depth discrepancy and the second penalizes foreground mask discrepancy. There is a threshold dm that determines the binary value of each pixel of the r_m dependant to the amount it differs between o_d and r_d. A small value ε is added to avoid division by zero and λ is normalization constant.

2.1.3 Optimization using PSO

Particle Swarm Optimization (PSO) is the algorithm that optimizes an object function through the evolution of atoms of a population. Every particle holds its current position in a vector xk and its current velocity in a vector vk. The vector Pk stores the position at which each particle achieved, up to the current generation k. the vector Gk stores the best position encountered across all particles of the swarm.

$$v_{k+1} = w(v_k + c_1 r_1(P_k - x_k) + c_2 r_2(G_k - x_k))$$

and

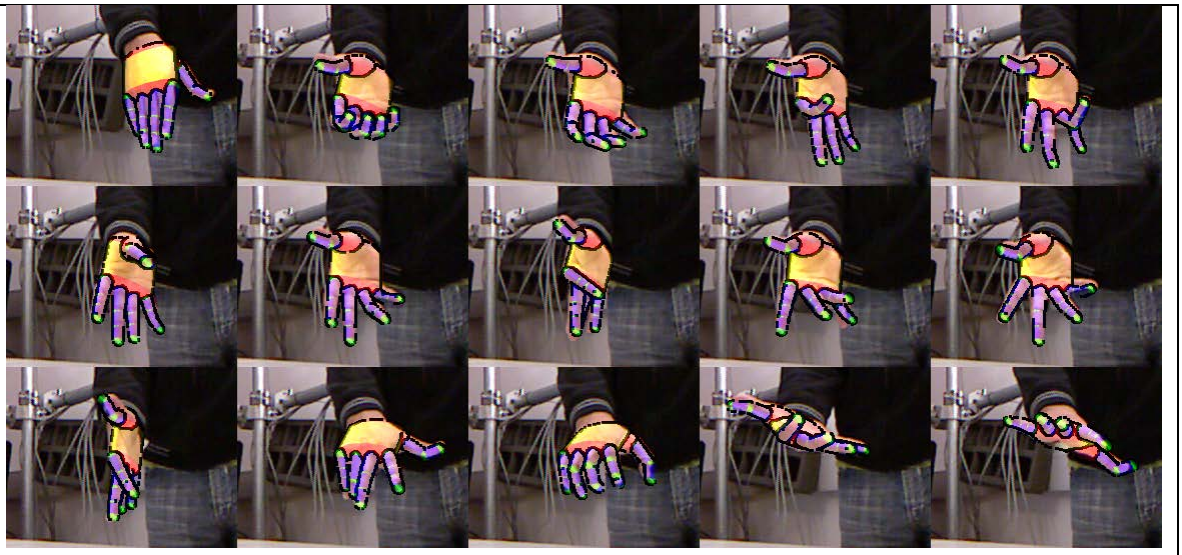
$$x_{k+1} = x_k + v_{k+1}$$

Knowing the velocity and the position of each particle the FORTH Hand Tracker estimates the optimum pose of the hand by calculating all the translation and the rotations of the 26 DOF hand model particles.

At every frame, the estimated hand pose is stored in the 27 parameters vector which can be encoded to the current hand pose and position. This 27 parameters vector we use to recognize the gestures the user performs.

2.1.4 Results

Indicative snapshots are shown in Fig. 3. As it can be observed, the estimated hand model is in very close agreement with the image data, despite the complex hand articulation and significant self occlusions.

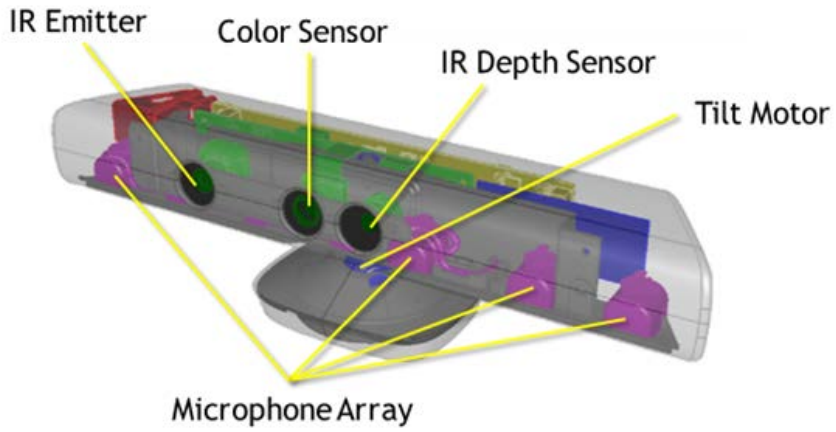


Picture 3. Indicative results on real-world data

2.2 Kinect Sensor

Kinect is a sensor created and first published by Microsoft as a side gaming devise for Xbox360 gaming console (11) (12). Its category named RGB-D (Red-Green-Blue-Depth) sensor. It combines several different devices working together. These are an RGB camera, a depth sensor, four microphones and a motorized tilt. The RGB camera provides a colorful image and it is used as a simple RGB camera. The difference with Kinect is the depth sensor that provides depth information for many of the items capture in 2D camera plane. The depth sensor consists of an infrared projector and an infrared camera. The infrared camera captures the reflections of the infrared waveform and provides information for the distance of the reflected items, if they are in capturing range. Together the RGB camera and the depth sensor are a powerful tool in Computer Vision science. The RGB image provides information for chromatic characteristics and for the template of an item. Combined with the distance from them, given from the depth sensor, can provide the width and the height of the item. Depth sensor provides the depth so we have the X, Y and Z of an item or a single point of the image in 3D space. The output of the depth sensor is a grayscale image, called depth map. It is same size with the RGB image and each pixel scale represents the distance from the camera. The darker points are closer to the camera. If a point is out of the range and the depth cannot be measured, the pixel in this point is black. The range that the sensor gives trusty depth information depends to the operation mode and it is 0.8 to 4 meters for the default mode and 0.4 to 3 meters for the near mode. Objects in greater distance cannot provide appropriate infrared reflections for measurement and nearest objects are not in the plane of view of the two cones of both infrared projector and infrared camera.

Picture 4. Kinect Sensor



Picture 5. Kinect Structure

Picture 6. Depth Map

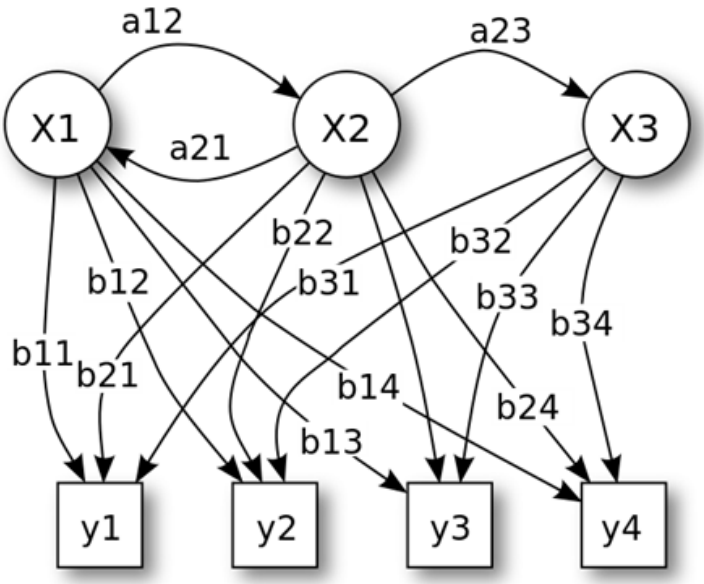
Although Kinect Sensor was first created for gaming, its low cost and wide availability raise the interest of many developers to create applications to interact with computers, to operate robotic systems, to provide help in medical applications, even to help people with movement or speech difficulties. Its software was in first embedded in Xbox console and had mostly to do with Skeletal Tracking and Facial expressions recognition. In February 2012 Microsoft released the Kinect Software Development Kit (SDK) for Windows for programmers to build their own applications, but many more programming tools are available. Today there are two versions of Kinect Sensor, Version1 and Version2 and several more RGB-D sensors from other manufacturers.

2.3 Hidden Markov Model (HMM)

Hand gestures recognition plays a crucial role in our application. The mathematical model we used to detect these gestures is the Hidden Markov Model (HMM) (13) (14)(15). HMM is a special occasion of Markov Model. Markov Model is a stochastic model used to model dynamic systems assuming that the future state depends on the current state only, not in all state history. The benefit in this in computational systems is the low need of computing power and the low time of processing. We have not to process all the state history but only the current state assuming that the current state is based in the previous and the previous in the state before it and so on. The Hidden Markov Model is a special occasion of Markov Model used when the state is not observed directly. It is commonly used in Human Computer Interaction in several applications such as speech, handwriting, gestures and expressions recognition.

The states of an HMM represent the un-observable internal configuration of the model. The observations are used to estimate the hidden states. In our case an HMM encodes a gesture that consists of one or more poses. The states of that HMM represent these hand poses. The observations in our case are Euclidian distances between specific parts of the hand. The "thumbs up" pose for example can be observed if the thumb fingertip is over a threshold away from the palm and all the other fingertips are under a threshold close to the palm.

For each frame all the distance features needed for each gesture are calculated and give a probability for each pose. These are called observation or emission probabilities. Each gesture is modeled as a separate HMM that models the poses of the gesture and the transition between the poses. The probability of transition between two poses is called transition probability. For a given frame sequence the posterior probability of each gesture is calculating using the emission and transition probabilities of the corresponding HMM. The HMM with the greatest probability represents the optimal gesture.



X — states
y — possible observations
a — transition probabilities
b — emission probabilities

Picture 7. HMM Example

In picture 7 it is an example HMM with 3 states and 4 possible observations. If we suppose that the previous state was X2 and now we observe the observation y3, the probabilities of the current state to be X1 or X3 are:

$$X1_{prob} = a21 * b13$$
$$X3_{prob} = a23 * b33$$

The state with the greatest probability is the optimal state at the current time.

2.4 Viterbi Algorithm

The Viterbi algorithm (16) is a dynamic programming algorithm which finds the most likely sequence of hidden states in a sequence of observed events such as Hidden Markov Models. It has many applications in CDMA and GSM digital cellular, dial-up

modems, satellite communications, wireless Lan transmissions and most recently in human activity recognition such as speech and gesture recognition. The predicted most likely sequence of hidden states through Viterbi Algorithm is called Viterbi Path. The maximum probability of a Viterbi Path is called Viterbi Probability. Viterbi probability is not always the probability of the optimal state at the current time, but the probability of the optimal path of states by the moment. The first Viterbi Probability is the probability produced from the Starting probability for each state and the Observation Probability for each state. The next Viterbi Probability is the product of the Transition Probability from each previous state to each current state, the Observation Probability of each current state and the Viterbi Probability of each previous state. So if at the first time period the state X1 had greater Viterbi Probability than the state X2, but at the second time period the total product of the Viterbi Probability of the X1 and the Transmission Probability from X1 to X3 and the Emission Probability of X3 is greater than the corresponding probability moving from state X1 to state X3 probability, the new optimal Viterbi Path passes through X2 and not X1 that had the greatest probability at first time.

$$V_{1,st} = Obs_{st} Start\_P_{st}$$

$$V_{t,st} = \max_{prev\_st} (Obs_{st} Trans\_P_{prev\_st,st} V_{t-1,prev\_st})$$

Where V is the Viterbi Probability, Obs is the Observation Probability, Start_P is the Start Probability, Trans_P is the Transition Probability, St is the Current State, Prev_st is the Previous State and T is the Time in program circles

Section

3

Implementation

3.1 Section Structure

In this section we represent the whole process of the implementation of the Gesture Human-Computer Interaction (GHCI) application. In the Camera Calibration section we describe how we calibrate the camera's coordinate system to the screen's coordinate system. Next in Cursor Movement we represent how we succeed the cursor movement according to the position we point to the screen with our hand. In the Poses and Gestures section we represent the gestures have been chosen and the poses consist them to interact with the computer. In Observations section we describe how we estimate when a pose is performed. In Design of HMM we model the desirable gestures, according to the transitions between poses. In Gesture Selection we describe how we calculate the probability of each gesture to been performed and we show how we choose the optimal gesture. In Modes of Operation we describe how we transit between the modes of operation and how each computer function is achieved in each one of them. Finally in the User Interface section we represent how the user operates the application and interacts with the computer.

In the next list we see the steps of process to interact with the computer.

1	Camera Calibration	Calibration from camera to screen
2	Cursor Movement	Movement of cursor on hand's position
3	Poses and Gestures	Definition of poses and gestures
4	Observations	Observations of poses
5	HMMs	Gesture models build
6	Viterbi Algorithm	Finding gestures probabilities
7	Gesture Selection	Choosing the optimal gesture
8	Computer Function	Translation of gestures to computer functions

List 1. Implementation process steps

3.2 Camera Calibration

Getting started we have an RGB-D camera operated with Python language GUI. We take images through GUI's visualization and the FORTH Hand Tracker attaches a 3D 26DOF hand model and tracks our hand on these images. The first operation of our application is that the mouse cursor is moving according to the hand's position with respect to a coordinate frame aligned with the computer screen. Due to this, we do the calculations to transform hand's position from the camera's coordinate system to the screen's coordinate system.

In our case the camera is set in side position in order to be able to have the screen in its field of view. The Y axis of the camera has the same orientation with the Y axis of the screen, while it is translated on screens Z axis. The Camera's X axis has almost the orientation of the screen's Z axis, while the screen's X axis has almost the orientation of the camera's Z axis. It is not necessarily to set the camera in this exactly position but after many tests we found it the most convenient due to its field of view that concludes the screen and a big enough area in front of it, in which the user can perform the gestures.

Picture 8. Camera positioning

The hand's position in the camera's coordinate system is taken in 3D space through the tracker's parameters where the first 3 values, correspond to hand palm's X, Y and Z position.

We compute the Rigid Motion transformation (17). To calculate the transformation that method requires at least 3 3D point correspondences. For that purpose we use the

coordinates of the four screen corners. We suppose the corners' coordinates in screen's system starting from the upper left corner and moving clockwise as below.

$$a = [0,0,0]$$
$$b = [w,0,0]$$
$$c = [w,h,0]$$
$$d = [0,h,0]$$

Where w and h are screen's width and heght

We take the same four coordinates in camera's system from the coordinates of the index fingertip when user touches each corner and presses the appropriate key [Calibrate the Camera]. We name them ac, bc, cc, dc. We calculate the centroids in two systems by adding the four coordinates and dividing by four.

$$\text{Screen Centroid} = \frac{a + b + c + d}{4}$$
$$\text{Camera Cendroid} = \frac{ac + bc + cc + dc}{4}$$

Having two sets of points we produce the projection matrix (H matrix) which maps the vector of response variable values to the vector of fitted values. We subtract each centroid from each corresponding point and we sum the products of the response values, in our case the camera points (ac, bc, cc dc) and the equivalent fitted values which are the screen point values (a, b, c, d), for each point. The result is a 3x3 square matrix we use to solve the Singular Value Decomposition (SVD). The product of the right-singular vectors (U) and the left-singular vectors (V) gives us the rotation (R) of the camera coordinate system from the screen coordinate system. The translation (T) from the camera coordinate system to the screen coordinate system is the abstract of the product of the rotation and the camera centroid from the screen centroid.

$$H = ((ac - CamCent) * (a - ScreenCent)') + ((bc - CamCent) * (b - ScreenCent)') + ((cc - CamCent) * (c - ScreenCent)') + ((dc - CamCent) * (d - ScreenCent)')$$
$$[U, S, V] = \text{svd}(H)$$
$$R = V * U'$$
$$T = -(R * CamCent) + (ScreenCent)$$

To do the calibration the camera must be in a position watching the screen. We choose a side position for the camera, so the screen and the space in front of it that the hand will

move, are in the sight of the camera. The left side looking to the screen has been chosen because it is easier to recognize the gestures when we use the right hand. If the camera was in a position not watching the screen the transformation and so the camera position should be prefixed or the user should done it, counting the distances from the camera center to the screen center, so it would be difficult and time consuming.

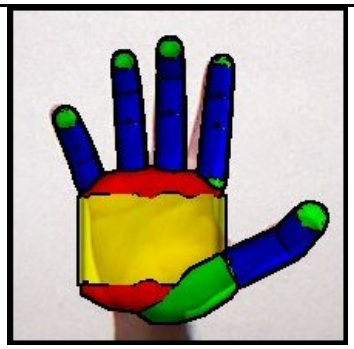
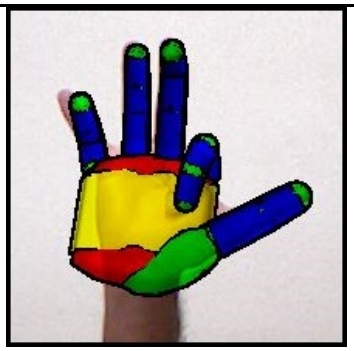
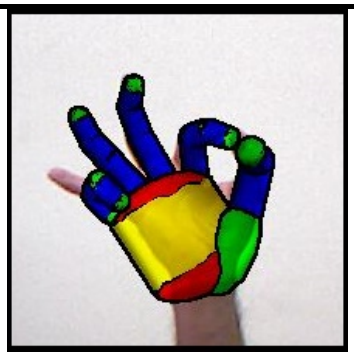
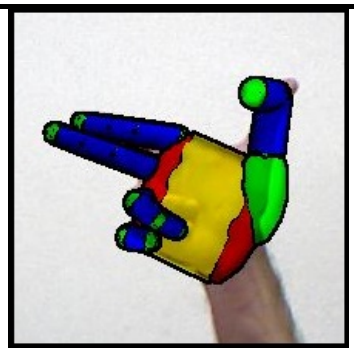
3.3 Cursor Movement

Cursor moves on the hand palm's coordinates which are produced by the FORTH Hand Tracker. It follows the X and Y position transformed in screens coordinate system. So the user moves the cursor by showing with his hand the position he wants on the screen. An alternative would be the cursor to follows the index fingertip's position for greater accuracy, but rejected through to relocation of the cursor when user attempts a gesture that uses the index finger. Besides that, using the PC from the suggested distance of a half meter, sitting comfortably on your desk chair, it is easier to aim taking the whole hand as a single point.

Hand's previous position is hold when hand's current position taken. They compared, demanding a relocation over one millimeter in both X and Y axis to fulfill the movement. This is a threshold used because of a default trembling of the hand model trying to guess the optimal hand's position. So cursor movement has an accuracy of one millimeter, which is good enough for the windows' icons, buttons and scroll menus. It is not ideal for designing applications, except if the zoom is big enough to correspond one pixel to one millimeter. The positioning of the cursor to the coordinates is implemented using freeware virtual mouse library.

3.4 Poses and Gestures

We have set six different hand poses which used to perform five different gestures that our application can recognize. Each gesture consists of two poses, one common to all and one unique for each one. As gesture we set the transition from one specific pose to another. The first pose consisted in a gesture is called starting pose and it is common to all gestures. The second pose is the unique pose and we call ending pose. In the picture 8 we see the six poses and at the list 2 we see the five gestures with those pose transitions.

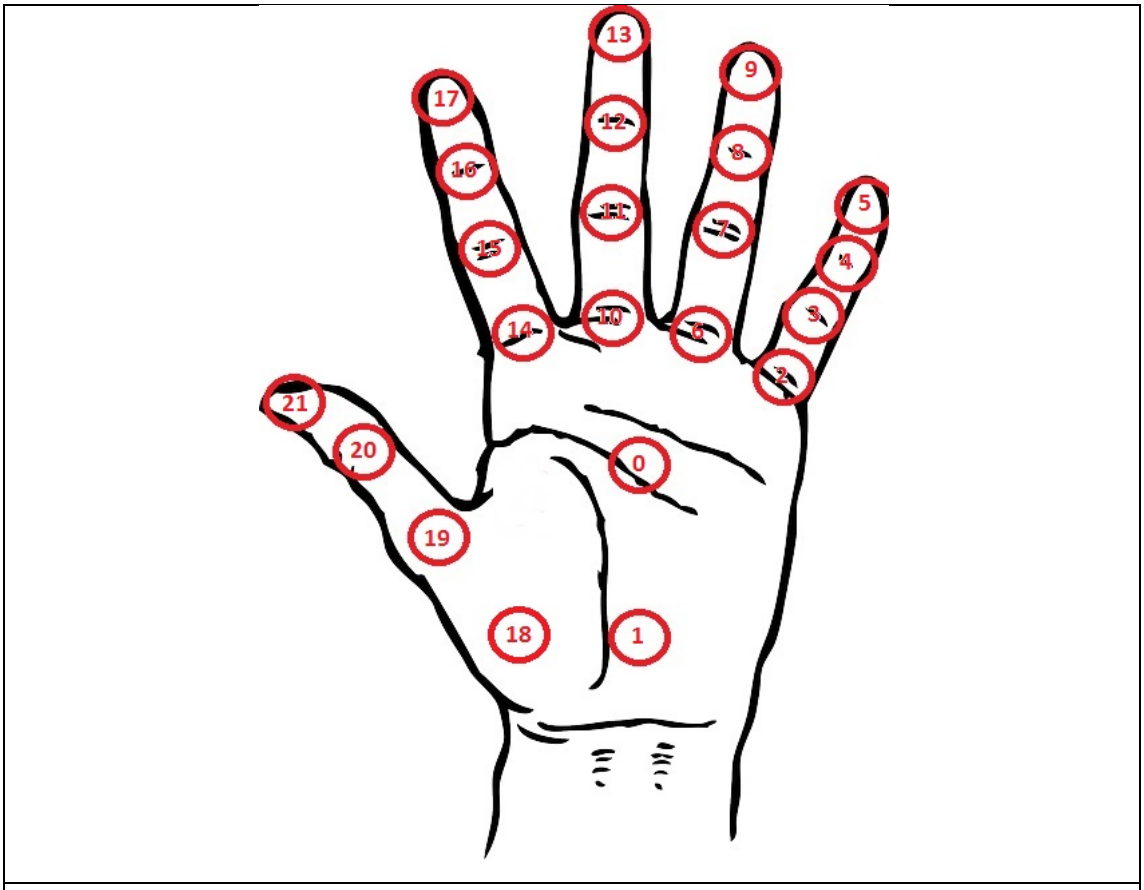
		
Open Hand	Index Down	Middle Down
		
OK!	Single-Barrel Pistol	Double-Barrel Pistol

Picture 9. HMM Example

Gesture	**Starting Pose**	**Ending Pose**
Left Click	Open Hand	Index Down
Right Click	Open Hand	Middle Down
Grab	Open Hand	OK!
Zoom	Open Hand	Single-Barrel Pistol
Volume	Open Hand	Double-Barrel Pistol

List 2. Gestures

To recognize each pose we use auxiliary function for the Hand Tracker which also built in FORTH, which gives us the 3D coordinates of 21 specific points on the hand model, two for the palm and four for each finger. The position of each point is shown in picture 9.



Picture 10. Separate points with known geographical position

We use the Euclidian distance between two or more of these spheres to recognize each hand pose. All poses are described from the distance of each fingertip from the palm, or from the distance of two fingertips between them or both. Pinky finger has been exempted from all poses because it is the most possible for the tracker to lose it, so it is not participate to the gestures. The first pose is the Open Hand where all fingertips are away from the palm. The Index Down pose is when index fingertip is close to the palm and all other fingertips are away from the palm. The Middle Down pose is when middle finger fingertip is close to the palm and all other fingertips are away from the palm. The OK! pose is when thumb fingertip touches index fingertip and all other fingertips away from the palm. When ring and middle fingertips are close to the palm and index and thump are away from each other is the Single-Barrel Pistol pose. When ring fingertip is close to the palm and middle and index fingertips are away from the palm and the thump fingertip is the Volume pose. Edges have not been used to recognize poses because they vary according to hand's rotation. The method with the distances between the specific points ensures that the pose is recognized easier at any rotation of the hand.

3.5 Observations

We have pre-defined the ideal distances of interest as follows.

Pre-defined Distances	Notation
Index close to the palm	$d_{index-down}$
Index away from the palm	$d_{index-up}$
Middle close to the palm	$d_{middle-down}$
Middle away from the palm	$d_{middle-up}$
Ring close to the palm	$d_{ring-down}$
Ring away from the palm	$d_{ring-up}$
Thumb close to the palm	$d_{thumb-down}$
Thumb away from the palm	$d_{thumb-up}$
Thumb touches index	$d_{thumb-index-t}$
Thumb away from index	$d_{thumb-index-a}$
Thumb away from middle	$d_{thumb-middle-a}$

List 3. Pre-defined Distances

In every program circle we count the Euclidian distances between points of interest as follows.

Counted Euclidian distances	Notation
Index - Palm	$eu_{index-palm}$
Middle - Palm	$eu_{middle-palm}$
Ring - Palm	$eu_{ring-palm}$
Thumb - Palm	$eu_{thumb-palm}$
Thumb - Index	$eu_{thumb-index}$
Thumb - Middle	$eu_{thumb-middle}$

List 4. Euclidian Distances

To observe a pose we sum all the differences of the counted Euclidian distances from the ideal pre-defined distances, who take place for this pose. Ideally it should be zero but that is never true, so the pose with the smallest summary of distances is the most likely pose.

Pose	Distances Summary
Open Hand	$D_{OH} = abs(d_{f-up} - eu_{f-palm})$
Index Down	$D_{ID} = abs(d_{index-down} - eu_{index-palm}) + abs(d_{f-up} - eu_{f-palm})$
Middle Down	$D_{MD} = abs(d_{middle-down} - eu_{middle-palm}) + abs(d_{f-up} - eu_{f-palm})$
OK!	$D_{OK} = abs(d_{thumb-index-t} - eu_{thumb-index}) + abs(d_{f-up} - eu_{f-palm})$
Single-Barrel Pistol	$D_{SB} = abs(d_{thumb-index-a} - eu_{thumb-index}) + abs(d_{f-down} - eu_{f-palm})$
Double-Barrel Pistol	$D_{DB} = abs(d_{thumb-index-a} - eu_{thumb-index}) + abs(d_{thumb-middle-a} - eu_{thumb-middle}) + abs(d_{ring-down} - eu_{ring-palm})$
where f is abbreviation for the rest of the fingers (except pinky which has been exempted)	

List 5. Distances Summaries Equations

We use the exponential function to bring the value which is closer to zero, to be closer to one, which is the greatest probability for the chance of that pose.

Pose	Observation probability
Open Hand	$Z_{OH} = e^{-\frac{D_{OH}^2}{2s^2}}$
Index Down	$Z_{ID} = e^{-\frac{D_{ID}^2}{2s^2}}$
Middle Down	$Z_{MD} = e^{-\frac{D_{MD}^2}{2s^2}}$
OK!	$Z_{OK} = e^{-\frac{D_{OK}^2}{2s^2}}$

Single-Barrel Pistol	$Z_{SB} = e^{-\frac{D_{SB}^2}{2s^2}}$
Double-Barrel Pistol	$Z_{DB} = e^{-\frac{D_{DB}^2}{2s^2}}$
List 6. Observations Equations	

Because of the ideally distances variety in different hand sizes and anatomy, we choose to use a comparative method to identify the optimal pose. Every program circle each pose is compared to the Open Hand pose and to all the other poses. The pose with the greatest probability, is the dominant pose.

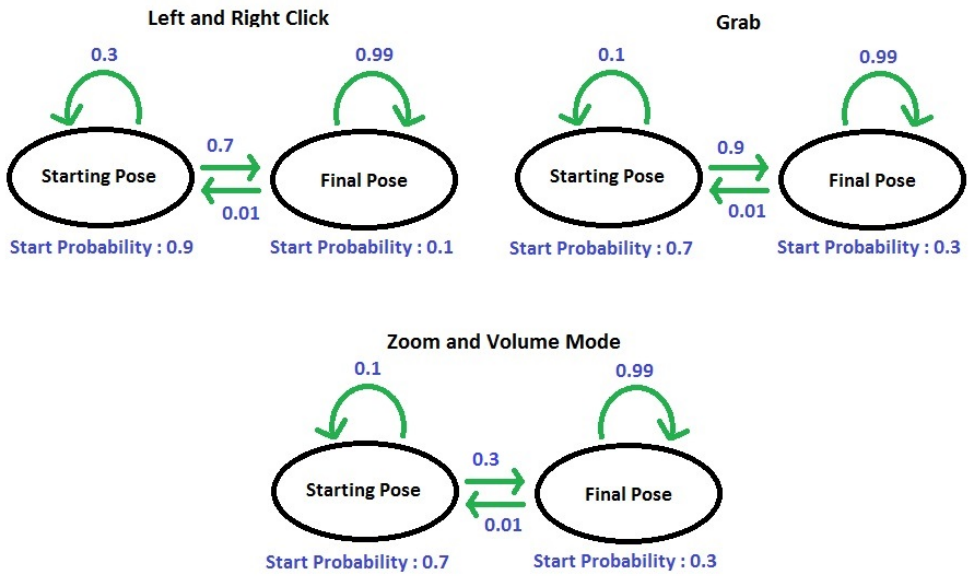
We did not count on a single observation, first to avoid instant errors and second to have an image of the whole gesture, not only the instant pose. So we keep a history of the last five images and we come to the optimal gestures through the sequence of the observed poses.

3.6 Design of HMM

The method we use to come out to the optimal gesture through the sequence of the last five observed poses is the Hidden Markov Models or HMMs. We have simplified as much as possible the gestures in order to be easily recognized with a prediction method as HMMs. All the gestures made up of two states that represent the two different poses. The starting state is common to all gestures and it is the pose of the Open Hand. The second state is for each gesture the ending pose for this gesture. All the ending states are clearly different from each other, so we do not have complex HMMs with the same ending and different paths through middle states. This is giving us the advantage of a fast and easy recognition with small chance of error. The disadvantage is the limited number of gestures can be recognized with this method. If we wanted to recognize much more gestures we should use more complex HMMs with more states.

We have built five different HMMs for the five different gestures we want to be recognized. The HMMs parameters are the starting state probability, the transition probability and the observation probability. The observation probability is the observed probability of each pose at the last five frames, as counted by the Euclidian distances. As the counted Euclidian distances approach the given distance for each pose, the exponential function of their differences approaches to one, which is the greatest probability.

The starting state probability and the transmission probabilities depend on the gesture. In two first gestures, left click and right click, we want the activated event to happen while the corresponding finger is moving down and to been completed while the finger reach the wanted distance. So the starting probabilities are a little bigger for the Left and Right Click when in Open Hand. Besides that they are the most usual functions, so we give a better chance the next gesture user wants to perform, when in Open Hand, to be one of them. At the other three gestures we want to implement the function when the ending pose has been performed. We want for example to activate Grab when thumb and index fingers touch each other and not when they moving close to each other. That also decreases the chance to mistake the Grab with the Left Click gesture, which both of them demand a down move of the index. The best starting state probability remains the Open Hand, but it is not such big as in Left and Right Clicks. In addition the starting state probability for the ending pose is bigger in these three gestures than it is in Left and Right Clicks. For the same reason the transmission probabilities are a little different for the same groups of gestures. In Left and Right Clicks the probability to transmit from Open Hand to Open Hand is a little greater than the other three gestures, that it is smaller. In Grab there is greater probability to transmit directly to the ending pose, because we want to activate the event when we are already there. In Zoom and Volume Mode we give smaller transition probability from the starting to the ending pose because they are gestures that change the mode of operation so we reduce the chance to activate them accidentally. In all cases the transmission to ending state is greater than to stay in Open Hand for a while. In all cases when in ending pose, the probability to remain in ending pose is much greater and the probability to go back to Open Hand which is almost zero. For all gestures, when we are in Open Hand again, the gesture has already been finished. We can see the probabilities by numbers in picture 11.



Picture 11. HMMs

3.7 Gesture Selection

To determine the optimal gesture according to HMMs probabilities and the observation probabilities we use the Viterbi Algorithm. The Viterbi Algorithm calculates the optimal sequence of states of an HMM. Since we have two states we are looking for paths that start from the Starting State and end to the Ending States. This can be happen to more than one HMM each time. For example when we perform the Grab gesture we have a path that starts with the Open Hand pose end ends with the OK! pose. At the same time the Left Click gesture as well has a path of states that starts from the Open Hand pose and ends with the Index Down pose because actually the index finger is a bit closed in the OK! pose. If we actually perform the OK! pose after the Open Hand pose, then the HMM that models the Grab gesture will have a total path with greater probability of that of the HMM that models the Left Click gesture. So in each program circle we calculate the Viterbi probabilities for all the HMMs/Gestures holding data from the last five frames. All these probabilities are kept in a list to compare to each other. The bigger value defines which is the gesture that the user performs at the moment.

V_{left_click}
V_{right_click}
V_{grab}
V_{zoom}
V_{volume}
List 7. Viterbi Probabilities List

$$\text{Best Probability} = \max(\text{Viterbi Probability List})$$

3.8 Modes of Operation

There are four modes of operation for the application. The initial starting mode is the normal mode. The other three modes are the Zoom Mode, the Volume Mode and the Calibration Mode. In Normal Mode all five gestures can be recognized. Three of them, the Left Click, the Right Click and the Grab, spontaneous activate the appropriate function on computer's operating system. The other two gestures, the Zoom Mode and the Volume Mode, change the Mode to the same name Mode of Operation. In the Zoom

Mode and Volume mode no gesture can be recognized, but only functions can be activated. In total we have eight different functions which can be activated.

Modes of Operation	Functions
Normal Mode	Left Click
	Double Click
	Right Click
	Grab/Ungrab
Zoom Mode	Zoom In
	Zoom Out
Volume Mode	Volume Up
	Volume Down
Calibration Mode	-

List 8. Available functions for each Mode of Operation

Some of the functions are activated spontaneous when the corresponding is gesture performed and some other are a combination of the gesture and its duration or the gesture and the positioning change of the hand or of parts of it.

functions	Activation Method
Left Click	Gesture spontaneous
Double Click	Gesture + Duration
Right Click	Gesture spontaneous
Grab/Ungrab	Gesture spontaneous
Zoom In	Gesture + Positioning
Zoom Out	Gesture + Positioning
Volume Up	Gesture + Positioning
Volume Down	Gesture + Positioning

List 9. Activation method of each Mode of Operation

3.8.1 Normal Mode

When in normal mode the function Left Click activated spontaneous when the gesture recognized and a counter starts to increase. After a few seconds if the gesture has not been changed the function Double Click is activated and the counter turns to zero. Since then no other function can be activated until the pose turns to Open Hand again. If the pose turns to Open Hand before the Double Click been activated the counter turns to zero waiting the Left Click gesture recognition again. For all the activations we use commands from freeware virtual mouse library. For the activation of the Left Click we use two commands, the first is the left mouse button down and the second the left mouse button up. For the Double click we use the commands of the Left Click twice. The function of the Right Click is activated spontaneous when the gesture been recognized and no other gesture can be activated before Open Hand been recognized again. For the activation we use the commands for right mouse button up and down in the row. The function Grab is activated spontaneous when the gesture been recognized by the virtual command of the left mouse button down. When the pose returns to Open Hand again from Grab the Ungrab function is activated by the command of the left mouse button up. No other function can be activated in the meantime.

3.8.2 Zoom Mode

When the Zoom Mode gesture recognized, a counter starts to increase. If after a few seconds the gesture has not change, the application enters into the Zoom Mode. The Zoom Mode is a different mode of operation and no function is activated immediately when we enter in it. As long as we stay in Zoom Mode none of the five gestures can be recognized. The only thing considered is the Euclidian distance between thumb and index fingertips. The other three fingers are no longer tracked down, in order to gain tracking speed and to spare system recourses. While the distance of the two fingertips increases the Zoom In command is given. While it decreases Zoom Out command is given. The process is made to remind the user the Zoom In and Zoom Out method in touch screens, besides we do not touch any screen but we move our two fingers in the air. When we first enter in Zoom Mode we keep in a variable the current distance between the two fingers as the default zoom distance. When the distance changes we subtract the new current distance from the default distance. If the difference is positive over a threshold, Zoom In command is given as many times as the difference is bigger than the threshold. If the difference is negative and bigger than the threshold in absolute value, Zoom Out command is given as many times as the distance is bigger than the threshold. At the end of each program circle the default distance is replaced by the current distance.

Function	Number of times activated
Zoom In	$Z_{in} = \frac{\text{current distance} - \text{default distance}}{\text{threshold}}$
Zoom Out	$Z_{out} = \frac{\text{default distance} - \text{current distance}}{\text{threshold}}$

The Zoom In and Out functions are activated by using commands from freeware virtual keyboard library. We simulate the Ctrl+ and Ctrl- method for Zoom In and Out, that works in many Windows applications including photo viewer and pdf readers. When we reach the desirable zoom we keep stable the distance between our fingers and we get out of the Zoom Mode by moving our whole hand 30 cm to the right. When we get out of the Zoom Mode or any other mode, automatically the application returns to Normal Mode and all gestures are readable again.

3.8.3 Volume Mode

The Volume Mode works very similar to the Zoom Mode. The gesture of the Volume Mode does not activate immediately any command on the computer's operation system, but enters the application in a different Mode of Operation, where Volume Up and Volume Down functions can be activated. As in Zoom Mode, when the gesture of the Volume Mode been recognized a counter starts to increase and if we keep the gesture for a few seconds, it enters in Volume Mode. In Volume Mode no gestures are recognized. The user can only turn up or down the volume by moving his hand up or down. When we first enter in Volume Mode the current geographical position of the hand is set as the default position. If we move our hand up over a threshold distance, the Volume Up command is given, as many times as the distance we move is bigger than the threshold distance. If we move our hand down, we activate the Volume Down command, as many times as the distance is bigger than the threshold distance. In the end of the program circle the new position of our hand is set as the new default distance.

Function	Number of times activated
Volume Up	$V_{up} = \frac{\text{default distance} - \text{current distance}}{\text{threshold}}$
Volume Down	$V_{down} = \frac{\text{current distance} - \text{default distance}}{\text{threshold}}$

So we activate the Volume Up and Down events by changing the position of our hand in Y axis. None of the fingers are tracked when in Volume Mode in order to spare recourses and to increase tracking speed. The Volume Up and Down events are activated by the virtual keyboard library again. This time we simulate the Vol+ and Vol- keys that advanced computer keyboards feature. In order to exit the Volume Mode we only have to move again our hand 30 cm to the right and the application returns to Normal Mode.

3.8.4 Calibration Mode

Calibration Mode is an auxiliary mode of operation and it is not activated by any gesture but from the keyboard. It exists only to help us calibrate the camera to our screen as described in Calibrate the Camera. When in Calibration Mode no gestures can be recognized and no functions can be activated.

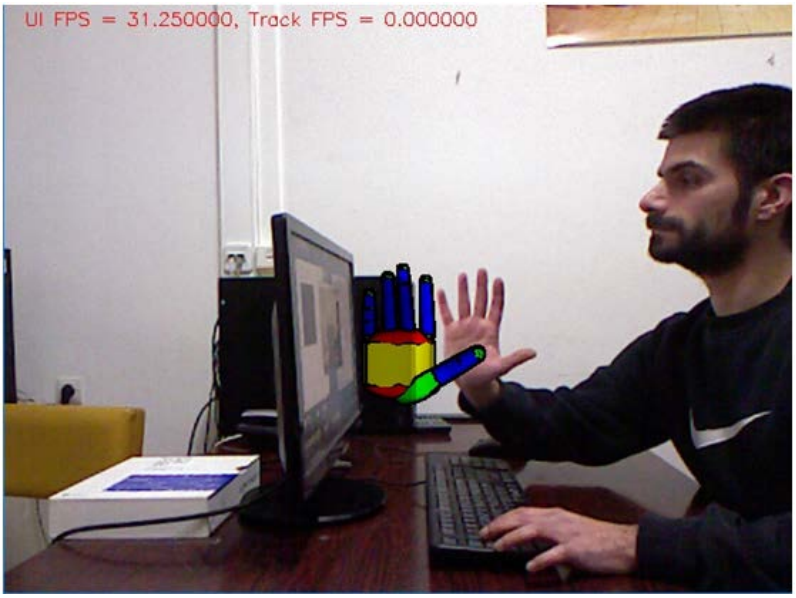
3.9 User Interface Overview

3.9.1 Hardware Setup

Before we start the application we shall put the Kinect sensor in a position able to see the computer screen and the area in front of it, where the hand will move and perform the gestures. The pre-calibrated position of the camera we use is 60 cm away on the left side of the screen, 10 cm in front of it and on the height of the middle point of the screen.

3.9.2 Start the Application

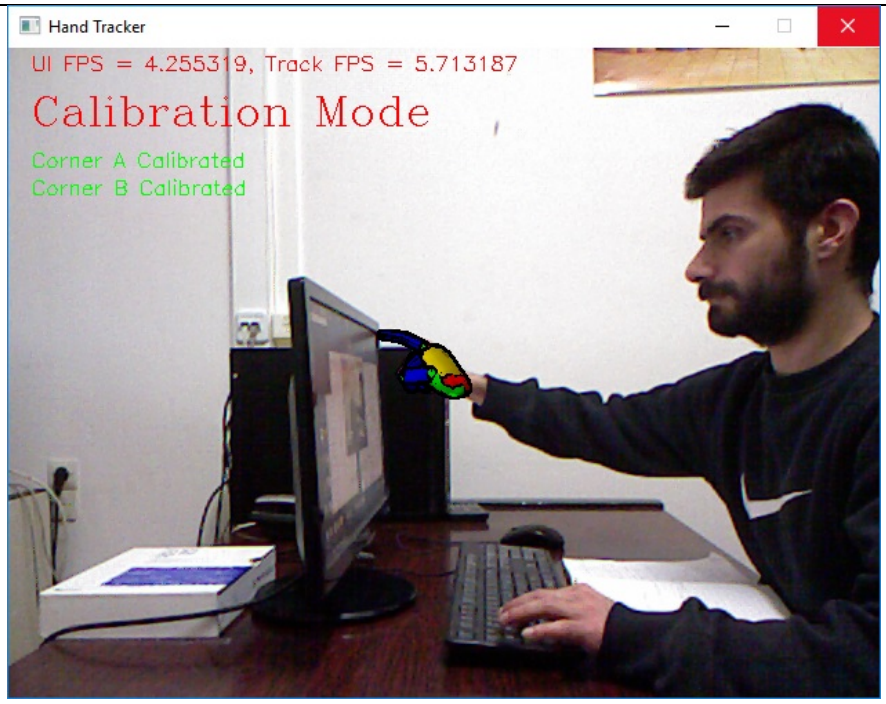
Starting the application a help video display window appeared, showing us the camera view and the model hand on it. The model hand is on a prefixed pose and position on the display. The position is on the middle of the display window and the pose is that of the Open Hand. The user must put his hand in the margins of the model acting the same pose to calibrate the model on his hand. When do so, by pressing the key “s” from the keyboard the application starts. The model follows the hands pose and position and the user can move the mouse cursor by showing with his hand the desirable position on the screen.



Picture 12. Starting the application

3.9.3 Calibrate the Camera

If the cursor does not follows hand's showing position, we must calibrate the camera's coordinate system on screen's coordinate system. The coordinate's calibration can be done by the user any time ruining the application. Pressing the key "k" from the keyboard, the application enters the Calibration Mode. The message "Calibration Mode" appears on the video display window. In this mode all other operations are disabled and the user shall touch the four screen corners, starting from the upper left corner and moving clockwise. Using the index fingertip, when touching the first corner (upper left), he presses the key "a" from the keyboard and then the keys "b", "c" and "d" for the other three corners respectively. Each time the key pressed a message that the corner has been calibrated is appeared. When all four corners are calibrated, automatically the program does the transformation and the cursor been attached to the position of the user's hand on screen's 2D plane. Pressing the key "x" the application exits the Calibration Mode and the main operations are enabled again. Each other program circle since then the application automatically transforms the hand position from the 3D space to 2D screen coordinates, due to latest calibration settings. If the camera or the screen has to change position, a new calibration shall be done.



Picture 13. Calibration Mode

3.9.4 Activate Left Click and Double Click

By performing the Left Click Gesture in Normal Mode the Left Click function is activated. We know we are in Normal Mode when there is no message says different in video display window. Staying in the final pose of the Left Click gesture for a while we activate the Double Click function. After that no other function can be activated until we return in Open Hand again.

3.9.5 Activate Right Click

By performing the Right Click Gesture in Normal mode the Right Click function is activated. No other function can be activated until we return in Open Hand pose again.

3.9.6 Activate Grab and Ungrab

By performing the Grab gesture in Normal Mode the Grab function is activated and stays active as long as we stay in the final pose of the Grab gesture. It works similar to holding the left click button of the mouse pressed. While do so among others we can select and move an icon, form a selection window to include multiple selected items or move a scroll bar. When we return in Open Hand again the Ungrab function is activated, similar

to releasing the mouse left click button. As long as Grab function stays active no other function can be activated.

3.9.7 Activate Zoom In and Zoom Out

By performing the Zoom gesture the application enters the Zoom Mode. It does not enter immediately but after a few seconds, due to a different Mode of Operation is activated. While in Zoom Mode the Zoom In and Zoom Out functions are enabled. The Zoom In function implements continually by distancing the index and thumb fingertips from each other. The Zoom Out functions implements continually by approaching the index and thumb fingertips to each other. Getting familiar with the application the user can estimate how much he has to distant or to approach his fingertips to activate Zoom In or Zoom Out functions only once. The method is designed to remind the way Zoom In and Zoom Out implement in a touch screen. To enter the zoom Mode we have to move our hand 30 cm to the right.

3.9.8 Activate Volume Up and Volume Down

By performing the Volume gesture the application enters the Volume Mode. Similar to the Zoom Mode it does not enter the Volume Mode immediately but after a few seconds. While in Volume Mode the Volume Up and Volume Down functions are enabled. The Volume Up command is given continually by moving the hand up. The Volume Down command is given continually by moving the hand down. The starting hand's position when user first enters the Volume Mode corresponds to the current Volume level. This can be any position in the camera's sight of view, but if we want to Volume Up and we enter the Volume Mode having our hand to high in the sight of view, we will not be able to move it up without camera lose it, so we must start from a lower position. Generally it is made the way that if the volume level is on fifty percent and we enter the Volume Mode setting our hand in the middle of the screen, we can implement a full Volume Up and Volume down move. To exit the Volume Mode we have to move our hand 30 cm to the right.

Picture 14. Volume Mode

3.9.9 Pause or Quit the Application

The application can be paused or un-paused any time while working by pressing the "p" key from the keyboard. The hand model freezes at its current position and pose which we must act again before un-pause again. To quit the application we press the key "q" from the keyboard.

3.9.10 System Requirements

The Gesture Human-Computer Interaction application is built on an Intel i5-3470 CPU system with 8GB installed RAM and an Nvidia GT-730 GPU. Minimum requirements are multi-core CPU with at least 1MB of installed RAM and CUDA-enabled GPU with 512MB of GPU RAM or more, supporting OpenGL 3.3 or later. The depth sensor is a Microsoft's Kinect for xbox 360. Operational System is Windows 10 pro 64bit with Visual C++ 64bit Redistributable Packages for Visual Studio 2013 installed. Sensor is installed using Prime Sense's Kinect SDK 5.1.6.6 and KinectMod 5.1.2.1 for windows 64bit drivers and handled with OpenNI 1.5.7.10 for windows 64bit image processing software. Programming language is Python 2.7.11 64bit.

Section

4

Experiments

4.1 Experimental Process

We tested the gesture based HCI application to a group of volunteers to assess its performance and functionality. The users were not familiar with the application or similar applications of HCI through gesture recognition. They had only 15 minutes time to test the application and then the experiment was started. In total 23 users took place to the experiments. Each one of them performed 5 times each of the 5 gestures which give us a total of 25 gestures for each user. Each experiment lasted about 10 minutes.

For the experiments all the mouse and keyboard events were disabled and the detected gestures were just registered. The information recorded with the registered event was the type of gesture, such as Left Click, and the time frame it was registered. The experiment supervisor was registering the actual performed (groundtruth) events. When the test finishes two lists are populated, one with the "Detected Gestures" recorded by the application each time a gesture was detected, and one with the "Groundtruth Gestures" registered by the supervisor each time a gesture was observed. The two lists are compared and if the type of the detected gesture and the frame it had been recognized matches with the type and the frame of the groundtruth gesture that been observed, it considered a successful recognition and the sample registered as True Positive. If the supervisor observe a gesture and record it but the application does not detect it, the event registered as False Negative. If the application detects an event that has not been actually happened, that is recorded as False Positive. So if the gesture event of the one category does not match in time frame with a gesture event of the same type from the other category the event is False and considered False Negative if the application does not detect it or False Positive if it detects it without it had been performed. If two events from the two different categories match in time but not in the type of the event, the application has confuses the gesture with another gesture and that is a False Negative for the one category and a False Positive for the other. If the application does not confuse the gesture but the event is a single False Negative or a single False Positive, that means that it failed to detect it in the first case, or that it confused a random hand movement with a gesture in the second case.

Because it is unlikely the observer to press the event key exactly at the same frame that the application detected it, there is a threshold of five frames before and after that the events considered matched. If two events match they marked as matched and they cannot been matched again even if another event of the same gesture is in the frame threshold. In that case the third event is considered as False Negative if it is from the Actual Event List or as False Positive if it is from the Detected Event List. All the other events that remained un-matched are considered as False Negative or False Positive respectively to the list they belong.

$$\text{TruePos} = \text{TruePos} + 1 \quad \text{if}(\text{frame}_{\text{detected event}} = \text{frame}_{\text{actual event}} \pm 5 \text{ and events are of the same gesture})$$

$$\text{FalseNeg} = \text{Unmatched events from Actual List}$$

$$\text{FalsePos} = \text{Unmatched events from Detected List}$$

When the true and false detections identified the Precision and Recall ratios are calculated. The Recall rates the mount of success according to how many gestures the application succeed to identify correctly, while the Precision rates the mount of success according to how well the application succeed not to been confused with a false detection.

$$\text{Precision} = \frac{\text{TruePos}}{\text{TruePos} + \text{FalsePos}}$$

$$\text{Recall} = \frac{\text{TruePos}}{\text{TruePos} + \text{FalseNeg}}$$

4.2 Results

We represent the results of the experiments showing the numbers of the True Positives, False Negatives and False Positives identifications, as well as the Precision and Recall ratios for each gesture, in a total population of 575 samples coming out from 23 users, each one of them performed 5 times each of the 5 gestures. We see the results for each gesture separately first and later the clustered results are represented. In the first column we see the absolute numbers of the True Positives, False Negatives and False Positives of the total population of the 23 users. In the second column we see the worst case while in the third column we see the best case of each statistic of a single user attempts. The worst and the best case are the absolute worst and best case for the stat and not the worst or the best user. In the Left Click for example the user that had 4 False Negatives is not

necessarily the same user that had 3 False Positives. In the last column we see the average score per user.

In Left Click gesture we had 94 True Positives, 21 False Negatives and 16 False Positives in the total population of 115 samples. The worst case of True positives was only one to be found in a single user while the best was all of the 5 recognitions to be correct. The worst case in False Negatives was 4 out of 5 while the best was no False Negative Identifications. In the False Positives the worst case of a single user was 3 False Positive Left Click identifications in the total time of 115 gesture performing attempts and the best case was no Left Click False Positives during all the experiment process. The average scores of the 23 users in Left Click gesture is 4.09 True Positives in 5 attempts, 0.91 False Negatives in 5 attempts and 0.7 False Positives during the whole experiment time. Those results give us a total 85% of Perception and 82% of Recall ratios.

	Left Click			
Stats	Sum (of 23 users)	Worst Case (of a single user)	Best Case (of a single user)	Average (per user)
Samples	115	5	5	5
True Positives	94	1	5	4.09
False Negatives	21	4	0	0.91
False Positives	16	3	0	0.7
Precision	-	25%	100%	85%
Recall	-	20%	100%	82%

List 10. Left Click Results

The Right Click scores were 109 out of 115 True Positive identifications, 6 False Negative identifications, while we had 11 False Positive Right Clicks in the total population of the

23 users. The worst case was 3 True Positives, 2 False Negatives and 3 False Positives for a single user while the best case was 5 True Positives and no Negative or Positive False for a single user. The average user score was 4.74 out of 5 True Positive recognitions, 0.26 False Negatives and 0.48 False Positive recognitions. The Precision and the Recall ratios were 92% and 95% respectively.

	Right Click			
Stats	Sum (of 23 users)	Worst Case (of a single user)	Best Case (of a single user)	Average (per user)
Samples	115	5	5	5
True Positives	109	3	5	4.74
False Negatives	6	2	0	0.26
False Positives	11	3	0	0.48
Precision	-	57%	100%	92%
Recall	-	60%	100%	95%

List 11. Right Click Results

In the Grab gesture we had 113 out of 115 correct identifications, only 2 loses, but we had 54 mistaken identifications. The worst case per user was 4 True Positives, 1 False Negative and 9 False Positives while the best was 5 True Positives and no False Negative or False positive identifications. The average score of the 23 users was 4.91 True Positives, 0.09 False Negatives and 2.35 False Positive identifications. The total score was 77% of Precision and 98% of Recall ratios.

In Grab gesture we notice a big amount of False Positives. This is due, first, to the closing movement of the index finger that is common to the Left Click gesture as well. The problem is not too big at the Left Click gesture because index has to almost touch the

palm but in the Grab gesture has to do less distance. If in the middle of the Left Click gesture the user has his thumb a little raised, depending to the angle camera sees him as well, the application may confuse it for Grab. However not all the False Positives are confusions from Left Clicks. Some of them are multiple recognitions in the same performance of the Grab gesture. If the angle changes while we performing the OK! pose of the Grab gesture, the tracker may see the two fingers separated for a while and after a second it sees them touched again, so it counts two Detected gestures in one Actual gesture. However the Perception ratio is on 77% which is the lowest score of all Perception and Recall ration in all gestures.

	Grab			
Stats	Sum (of 23 users)	Worst Case (of a single user)	Best Case (of a single user)	Average (per user)
Samples	115	5	5	5
True Positives	113	4	5	4.91
False Negatives	2	1	0	0.09
False Positives	54	9	0	2.35
Precision	-	38%	100%	77%
Recall	-	75%	100%	98%

List 12. Grab Results

In the Zoom gesture we had 111 out of 115 True Positives, only 4 False Negatives and only 1 False Positive. The worst score per single user was 4 True Positives, 1 False Negative and 1 False Positive while the best score was 5 True positives and no False Negatives or False Positives. The average score per user was 4.83 out of 5 True Positives,

0.17 False Negatives and 0.04 False Positives. The Precision ratio is 99% and the Recall is 97%.

	Zoom			
Stats	Sum (of 23 users)	Worst Case (of a single user)	Best Case (of a single user)	Average (per user)
Samples	115	5	5	5
True Positives	111	4	5	4.83
False Negatives	4	1	0	0.17
False Positives	1	1	0	0.04
Precision	-	80%	100%	99%
Recall	-	80%	100%	97%

List 13. Zoom Results

The Volume stats were 114 True Positives, 1 False Negative and 7 False Positives in the total population of samples. The worst score per single user was 4 True Positives, 1 False Negative and 2 False Positives. The best was 5 True Negatives and no False Negatives or False Positives. The average score was 4.96 out of 5 True Positives, 0.04 False Negatives and 0.3 False Positives. The Precision and Recall ratios were 95% and 99% respectively.

	Volume			
Stats	Sum (of 23 users)	Worst Case (of a single user)	Best Case (of a single user)	Average (per user)
Samples	115	5	5	5
True Positives	114	4	5	4.96
False Negatives	1	1	0	0.04
False Positives	7	2	0	0.3
Precision	-	71%	100%	95%
Recall	-	83%	100%	99%

List 14. Volume Results

The Zoom and the Volume gestures have the greatest scores because they are not been activated immediately but after a few seconds, as mentioned due to they lead to a different Mode of Operation. These few seconds give the time to the application to identify better the gesture avoiding the chance to confuse it instantaneous with some other gesture.

Generally the total scores are very high in most cases and it is more encouraging that we had total success per user in the best case at all stats in all of the five gestures, while we had no total failure in any gesture in the worst case. It is matter of fact that the users have to gain some familiarity with the application in order to use it properly, especially in which angle they shall have to their hands for each gesture. It is been observed that the most users gain this familiarity easily.

In the list 15 we see the Confusion Matrix of the five gestures in the total amount of gesture attempts of the 23 users. The vertical list to the left contains the actual gestures that the experiment supervisor saw with his eyes. The horizontal list contains the

detected from the application gestures. As we can see there is not much confusion between the gestures except from the Left Click gesture that had been confused many times with the Grab gesture. In addition there are not many failures of no detection at all of actual gestures. In the other hand there are bigger numbers of no actual gestures that detected mistaken as a gesture. The biggest problem is in the no gesture detected as Grab which is partially due to the multiple identifications in a single gesture attempt as we metioned.

Confusion Matrix							
		Detected gesture					
		Left Click	Right Click	Grab	Zoom	Volume	No Gesture
Actual Gesture	Left Click	**94**		19			2
	Right Click		**109**				6
	Grab	1		**113**			1
	Zoom		3		**111**		1
	Volume					**114**	1
	No Gesture	15	8	35	1	7	**-**

List 15. Confusion Matrix

Collecting the total scores of all users in all gestures we see that we have in the total population of 575 samples, 541 True Positives, 34 False Negatives and 89 False Positive Recognitions. The total Precision ratio is 90% and the total Recall Ratio is 94% for all the experimentation process.

Stats	Gestures					
	Left Click	Right Click	Grab	Zoom	Volume	Total
Samples	115	115	115	115	115	575
True Positives	94	109	113	111	114	541
False Negatives	21	6	2	4	1	34
False Positives	16	11	54	1	7	89
Precision	85%	92%	77%	99%	95%	90%
Recall	82%	95%	98%	97%	995	94%

List 16. Total Results

Section

5

Conclusion

To conclude we use the "FORTH Hand Tracker" to get information about the position of the human hand in the 3D space, from optical data that they are provided by the Kinect RGB-D sensor. We use the hand's position aligned to the computer screen's coordinate system to move the cursor in the position the user points on the screen. Furthermore several gestures are recognized which activate specific computer mouse and keyboard commands when they performed by the user. The hand gestures are modeled using the HMM where each gesture represented from a separate HMM. All the gestures are consisted of hand poses that correspond to the different states of the HMM. The poses are observed by calculating the Euclidian distances between individual hand parts. The probability of each gesture modeled as an HMM is calculated with the use of the Viterbi Algorithm. The gesture with the greatest probability defines which computer function will be activated. All the commands are categorized among their usage and they are consisted in three separate Modes of Operation. The mouse commands such as the Left Click, the Right Click, the Double Click and the Grab and Ungrab commands are consisted to the Normal Mode, the zoom commands such as the Zoom In and Out commands are consisted to the Zoom Mode and the volume control commands such as the Volume Down and Up commands are consisted in the Volume mode. Furthermore there is one more auxiliary mode for the camera calibration, the Calibration Mode.

The results of the experiments show the efficiency of the application especially if we consider that all the users had never use the application or similar applications before. Most of the users found it easy enough to operate the computer with this application, while they enjoyed the process.

5.1 Future Work

1. **Learning method for the different hand types.** We noticed that the difficulty in the use of this application varies among the different hand anatomies and that is because the thresholds of the Euclidian distances that are used to observe the gestures, were defined by the application builder's hand, making people with much different hand anatomy to find it more difficult to operate the application.

This could be solved by the use of some learning method, taking data from many different hand types, which would gave the ability to the application to change the thresholds according to the anatomy of the hand of each user.

2. **Left hand support.** Some of the users were left-handed and they found it a little harder to manipulate the supported operations with the right hand which is the only one supported by the application. So another upgrade could be the support of the left hand as well which the user could be able to choose when he starts the application.
3. **Training method for the HMM design.** In this version of the application we set the transition probabilities for the HMMs empirically because of the lack of dataset needed for the training method. If we had a dataset of people performing the wanted gestures, we could train the HMMs to find the best transition probabilities between their states, in our occasion between the poses.

References

1. https://en.wikipedia.org/wiki/Computer_vision.
2. **Ashwini M. Patil, Sneha U. Dudhane, Monika B. Gandhi, Nilesh J. Uke.** Cursor Control System Using Hand Gesture Recognition. *International Journal of Advanced Research in Computer and Communication Engineering.* 2013.
3. **Park, Hojoon.** A Method for Controlling Mouse Movement using a Real-Time Camera. *Brown University, Providence, RI, USA, Department of Computer Science.* 2008.
4. **Victor Adrian Prisacariu, Ian Reid.** 3D hand tracking for human computer interaction. *Image and Vision Computing.* 2011.
5. **Geovane Griesang, Rafael Peiter, Rolf Fredi Molz.** Man-Computer Interaction System Using Kinect. *International Conference on Industrial Engineering and Operations Management.* 2013.
6. **Maged N Kamel Boulos, Bryan J Blanchard, Cory Walker, Julio Montero, Aalap Tripathy and Ricardo Gutierrez-Osuna.** Web GIS in practice X: a Microsoft Kinect natural user interface for Google Earth navigation. *International Journal of Health Geographics.* 2011.
7. **Antonis A. Argyros, Manolis Lourakis.** Vision-Based Interpretation of Hand Gestures for Remote Control of a Computer Mouse. *European Conference on Computer Vision.* 2006.
8. **Iason Oikonomidis, Nikolaos Kyriazis, Antonis A. Argyros.** Efficient Model-based 3D Tracking of Hand Articulations using Kinect. *British Machine Vision Association.* 2011.
9. **Alexandros Makris, Nikolaos Kyriazis, Antonis A. Argyros.** Hierarchical Particle Filtering for 3D Hand Tracking. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.* 2015.
10. **Kennedy, James.** Particle Swarm Optimization. *Encyclopedia of machine learning.* 2011.
11. **Beleboni, Matheus Giovanni Soares.** A brief overview of Microsoft Kinect and its applications. *Interactive Mulitmedia Conference, University of Southampton.* 2014.
12. **Veisali, Eleni.** *Tele-operation of anthropomorphic robotic hand using optical data.* 2016.
13. **Lussier, Eric Fosler.** Markov Models and Hidden Markov Models. A Brief Tutorial. *INTERNATIONAL COMPUTER SCIENCE INSTITUTE.* 1998.
14. **Rodrigo Cilla, Miguel A. Patricio, Jesus Garcıa, Antonio Berlanga, Jose M. Molina.** Recognizing Human Activities from Sensors Using Hidden Markov Models Constructed by Feature Selection Techniques. *Algorithms.* 2009.
15. https://en.wikipedia.org/wiki/Hidden_Markov_model.
16. https://en.wikipedia.org/wiki/Viterbi_algorithm.
17. **Rabinovich, Olga Sorkine-Hornung and Michael.** Least-Squares Rigid Motion Using SVD. *Technical notes.* 2009.