A FRAMEWORK FOR DESIGN THINKING OF GESTURE-BASED HUMAN COMPUTER INTERACTIONS

KELVIN WAMBANI SIOVI

MASTER OF SCIENCE

(Information Technology)

JOMO KENYATTA UNIVERSITY OF

AGRICULTURE AND TECHNOLOGY

2021

A Framework for Design Thinking of Gesture-Based Human Computer Interactions

Kelvin Wambani Siovi

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Information Technology of the Jomo Kenyatta University of Agriculture and Technology

2021

DECLARATION

This thesis is my original work and has not been presented for a degree in any other University

Signature..... Date…….

Kelvin Wambani

This thesis has been submitted for examination with our approval as the University Supervisors.

Signature ..... Date .....

Prof. Cheruiyot W.K, PhD

JKUAT, Kenya

Signature ..... Date .....

Dr. Agnes Mindila, PhD

JKUAT, Kenya

DEDICATION

To my parents, mother Juliet and my late dad Prof. Elphas Wambani for making all that I do in life remunerative!

ACKNOLWDGEMENT

I would like to thank Sharon Wangari from the Nairobi Design Institute (NDI) for her in-depth works on various design thinking concepts and insights which motivated my interest in the field with the extra materials she provided.

I am very grateful to my supervisors Prof. Cheruiyot. Kipruto and Dr. Agnes Mindila for their invaluable guidance and support in assisting me with the entire research. They have been with me through high and low moments I experienced during research. I am very grateful to them and may God bless you.

Special acknowledgement to Hall Kirkham, who gave everything up when I started this degree and has never left my side, if only in spirit. Finally, to my parents; Elphas and Juliet for their support and encouragement throughout my studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

ABSTRACT

Gesture is a way of communication which involves body language and can be defined with or without spoken words. Gesture specifically is expression of thought or feeling where a degree of voluntarism is always implied by the body. If someone starts body activity at a sudden explosion, or bursts out laughing at something said, or if, on being told bad news, tears well up, these expressions are not usually regarded as gestures. The primary goal of gesture interaction applied to Human-Computer Interaction (HCI) is to create systems, which can identify specific human gestures and use them to convey information or controlling devices. Gesture interaction is already a promising input modality in smart computing environments such as modern gaming, augmented reality and virtual reality. Motion-based game controllers such as the Microsoft Kinect, Google Assistant make full body gesture interaction affordable and practical. The goal of gesture interaction is to computationally analyze body movements and associate each gesture to a predefined label. Most approaches are designed to control gesture events one by one given gesture till all are complete. Unlike touch input, users usually have little experience with gesture interaction. Technologists frequently claim that gestural input makes computing more "natural" by enabling communication with a computer the same way we also communicate with one another. Computer users face challenges when engaging and implementing gesture interactions, that is: complex coordination of all gestures regularly, mastering approachability to different gestures, muscle fatigue and a lack of supporting data input that is seamless with gesture interaction. In combination with a lack of tracking success or failure, end users often struggle to execute gestures correctly. To compensate this, users require an efficient design thinking framework that supports them during execution of gestures. Design thinking is a technological process guided by principles that provide a simple solution for a complex problem. An experimental research design using simulation tested the design thinking framework for using hand palm gesture interaction dataset. The framework classified hand palm gesture input data into its feedback, analytics and output entities in two steps: learning step (training step) in which a classifier is built to describe a predetermined set of new input data classes from the three entities. The second step builds a model done in the first training stage for classification with a classifier accuracy of known data to provide new alternative gesture interaction input data. The value of the design thinking framework results shows an improvement of gestural interaction improvement of real-time alternative gesture input sets for users to quickly learn how to interact with gestures systems for better gestural interaction performances.

CHAPTER ONE

INTRODUCTION

1.1 Introduction

When humans were evolving, there was an increasing demand of communication. Humans developed two different communication means; gestures were particularly important to communicate while chasing animals, allowing a silent communication that would avoid being perceived by animals; at the same time, to satisfy the increasing necessity to communicate information, humans developed articulated languages, which allowed continuous voice communication even while performing manual activities. Some theories suggest that first spoken languages originated from gestures and then evolved together in the different cultures. Several studies demonstrated that still nowadays humans use the same cognitive processes for the generation of speech and gestures (Angelini, 2017).

At the beginning of the digital information era, the interaction between humans and computers was reserved to few experts, who were acquainted with the low-level programming languages and command line consoles. Gesture recognition is a natural way of human computer interaction and an area of very active research in computer vision and machine learning. So, the primary goal of gesture recognition research applied to Human-Computer Interaction (HCI) is to create systems, which can identify specific human gestures and use them to convey information or controlling devices (Trigueiros, 2013).

The advent of touchscreen technology allowed the simplification of access to digital information also for older adults. Simple gestures, such as tapping, swiping and pinching allow to interact intuitively with the Graphical User Interface (GUI) and to manipulate directly virtual objects represented in the GUI. In other cases, it is referred to natural interaction whenever the humans' innate abilities are exploited.

Voice and gesture interaction with computer had been explored since the beginning of the computer era. Indeed, in 1980, Bolt's Put-That-There system already allowed to interact

with digital information through gestures and voice commands. Although voice could be seen as a very natural means to interact with computer, there are many situations in which voice recognition is effective (in noisy environment), or simply not acceptable for the user (in public or crowded environments). Many mobile and desktop operative systems integrate voice commands and vocal assistants (e.g., Apple Siri1, Google Voice2 and Microsoft Cortana3) but their usage is still limited.

1.1.1 Background

A gesture is a form of non-verbal communication in which visible bodily actions communicate particular messages, either in place of speech or together and in parallel with words. Gestures include movement of the hands, face, or other parts of the body. Human gesture is a mode of nonverbal interaction medium and can provide the most intuitive, originative and natural way to interact with computers. Gestures have been inspired by sign language can be used to offer simple commands for a computer interface. This gradually evolved with the development of much accurate infrared cameras and even fiberoptic bend-sensors that identify gesture input from users and provide desired output.

Human computer gesture recognition is a way for computers to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces graphical user interfaces, which still limit the majority of input to keyboard and mouse. Gesture recognition is technology that uses sensors to read and interpret hand movements as commands. In the automotive industry, this capability allows drivers and passengers to interact with the vehicle usually to control the infotainment system without touching any buttons or screens.

There are computing domains where gesture interactions support more flexible user performance while using digital devices. Digital devices range from the most common smartphones to complex computing machines like super computers. Gesture human computer interaction is spanning its wings across various industries like automobile, healthcare, media etc. Solutions available today are targeting specific market area. The

leading technology companies are offering solutions which cater to the need of specific gesture interaction. (Muser, 2015).

Applied design thinking methodology is a user-centered approach which draws on both the creative and rational thinking of multi-disciplinary teams to develop better solution to complex problems. There should always be a decrease in the amount of users' effort, how they should interact with technology. This is referred to as user investment. Design thinking provides a human-centered view of technological artifact design and is there-fore an appropriate tool for serious game design (Owen, 2006).

1.1.2 History of Computer Gesture Interaction

In the seventeenth century gesture was seen as a possible universal language and as the form in which language first arose. In the twentieth century interest in gesture declined, but was revived from about 1980, when it became of interest for cognitive psychology, the study of language acquisition, and for its role in communication in co-present interaction. Gesture study is thought to be relevant for understanding symbolic expression and conceptual processes. Its intimate involvement with speaking has implications for conceptions of language (R. P. Sharma & Verma, 2015).

Advancement with innovating for computer gesture Interactions massively increased with the availability of portable computing devices in early 1980's. Touch screen technology was becoming available for manufactures. The first personal computer featuring a touch sensitive screen allowed users to position the cursor and select on-screen buttons was developed in (1983) by Bell factories in South Koreas. It was one of the first consumer devices sold with an auditory sensor that received a patent for linking with a device using gestural interface.

The IBM's Simon personal communicator was the first attempt at a commercially viable smartphone, featuring a touch screen and a pager in 1994. The smartphone gained popularity and had the ability to allow user to hover their hands around fax machines which

would be able to incorporate the hovered words into fax machines'. A touch screen aided in the capturing of body movements

Human gesture interaction breakthrough was seen in the mid-1990s when the world wide web had begun to experience traction. Innovation in the gestural field increased too. A hand gesture interface system based on a pair of cloth gloves containing electrical sensors in each fingertip allowed developers and users of immersive visualization applications to easily and effectively interact with virtual environments was invented in1995. The innovation become a fun example of an Open Frameworks application that is sniff, with an interactive storefront window display discerning human behavior and engaging them in play.

A conceptual model by (Fritschi et al., 2004) for an intuitive gestural interface is incorporated for the development of web console devices and kinetics to enhance initial data gloves. Wii- Nintendo of Microsoft (2006) unveils a new video game console that accommodates both traditional buttons and physical gestures. Microsoft announced a multi-touch product Microsoft surface that combines software and hardware to offer image manipulation through hand gestures and physical objects in (2007). The same year Rectrix developed a gesture-based interface allows users to interact with interfaces using the movement of their limbs.

1.1.3: Gesture Interaction with Touchless Interfaces

Multi-touch interfaces like apple's iPhone and, Microsoft's Surface seem to represent the likeliest hardware touch in the near future. As gesture-recognition technologies come to fruition, other touchless interface technologies are quickly taking their place in the realm of too futuristic to be possible. One of the most eagerly anticipated is nonfiction headsets that allow computers to be controlled with thought alone (Charachon, 2013).

There are many reasons for using gestures as user input in human computer interaction. The most obvious ones are those based on context- specific or domain-specific requirements: For example, for a screen showing medical images in a sterile operating theatre or for a non-touch public display, gestural UIs with waving or pointing are a natural choice, because touch input has to be avoided or is impossible. Another example is the games domain for which Microsoft's Kinect demonstrates how gestural and full-body interaction of multiple co-located players can entirely change the players' experience and introduce a novel source of fun and motivation into gaming.

Touchless technology is "another layer" of control that integrates mental responses into other kinds of interfaces other than being stand-alone. That is trying to use a headset to replace a mouse, would lack accuracy and result in a 150-millisecond delay-too long for most users. The camera-based technology works by tracking a spot of infrared light reflected in a user's eyes to determine the location of his or her head. Then it measures the location of the eye's pupil and uses it to guide a pointer around the computer's display.

Currently as users, we only poke our finger at a mouse or keyboard, but eye contact is a huge source of context about our interaction and attention. Users can allow a computer to know where we're looking, it really opens up a whole new dialogue between computers and humans. User interface of a computer system is a language medium that enables interacting with a computer means that users can converse with the system and tell it what to do.

Just imagine how cumbersome it would be to drive a car from the backseat by having to tell the driver about every necessary action. This becomes even more cumbersome, if users have to learn the vocabulary and language of the driver. In the modern era of ubiquity, the principle of direct manipulation has extended beyond the boundaries of mobile devices.

1.1.4: Classification of Gesture based Human Computer Interactions

In inferring the intent from a human's gesture, it is helpful to have a classification of which type of gesture is being observed by (Nehaniv, 2005). The figure 1.1 below shows a

tentative classification of gestures based on application domains, enabling technologies, system response and styles.

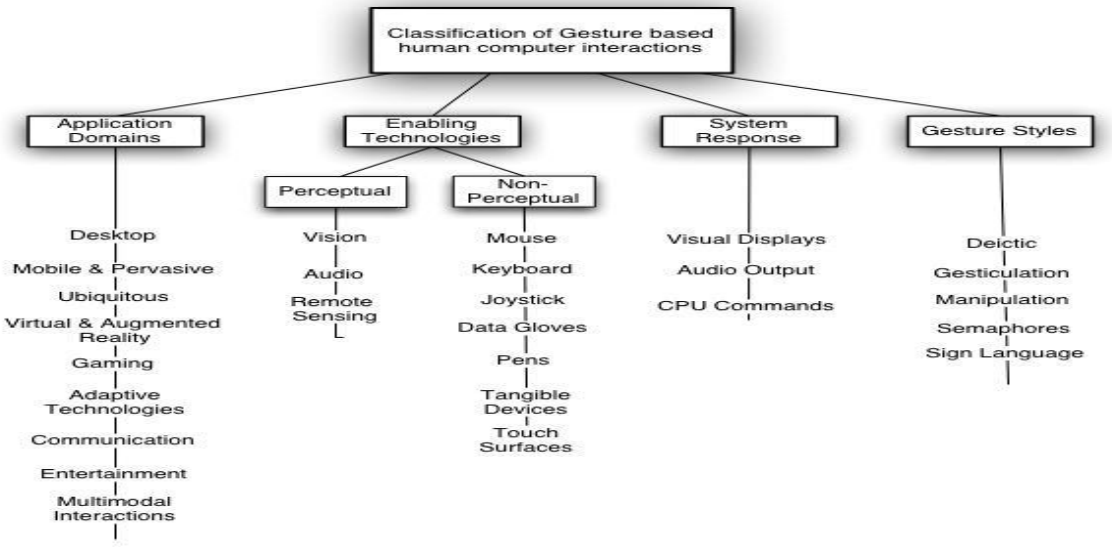


Figure 1.1: Classification of Gesture based Human Computer Interactions

The complexity of gesture in the context of situated human interaction depends on the classes of gesture styles described by the figure 1.1 above that provides a broader level of description and the steps towards a pragmatic criterion that are mostly used during gesture interaction, both physically and with a digital device.

i. Deictic (Pointing) – These are gestures that involve pointing the ident of an language-based (sign language) – They combine to form grammatical structures for conventional interfaces.
ii. Manipulative – These are gestures whose intended purpose is to control an entity by applying tight relationships within movements.
iii. Semaphores- Gestures employing a stylized dictionary of static or dynamic arm/hand gestures.
iv. Gesticulation – Gestures that moves in a way to communicate feelings to gain intent to bring attention.

1.1.5: Types of Gestures Recognitions

Gesture recognition enables humans to interface with the machine **(**HMI**)** and interact naturally without any mechanical devices. In gesture recognition technology, a camera reads the movements of the human body and communicates the data to a computer that uses the gestures as input to control devices or applications (Morris, 2000).

The following are different gesture recognition approaches applied in the context of human beings:

i. Hand Gesture Recognition: Hand gesture recognition acts as a highly adaptive interface between machines and their users. Hand gesture recognition technology allows operation of complex machines using only a series of finger and movements thereby eliminating for physical contact between operator and machine.
ii. Vision Gesture Recognition: There are two approaches to vision-based gesture recognition: Model based techniques: They try to create a three-dimensional model of the user's hand and use this for recognition. Image based methods: Image-based techniques detect a gesture by capturing pictures of a user's motions during the course of a gesture.
iii. Sign Language Recognition: Just as speech recognition can transcribe speech to text, certain types of gesture recognition software can transcribe the symbols represented through sign language into text. These measurements can be used to measure the distance of a human hand or other body part from an object; this facilitates a vast range of applications for a wide range of industries.
iv. Virtual controllers: For systems where the act of finding or acquiring a physical controller could require too much time, gestures can be used as an alternative control mechanism. Controlling secondary devices in a car, or controlling a television set are examples of such usage.
v. Remote control: Through this type of gesture recognition, the wave of a human hand is allowed to control various devices.

vi. Electrical Field Recognition: Proximity of a human body or body part can be measured by sensing electric fields. These measurements can be used to measure the distance of a human hand or other body part from an object; this facilitates a vast range of applications for a wide range of industries.

1.1.6: Major Applications of Gesture Interactions

The rising demand for mobile and computing devices enabled with innovative gesture recognition features will have a major impact on the overall development of the global gesture recognition market (R. P. Sharma & Verma, 2015). The following are major applications of gesture interactions.

i. Virtual Reality

   Virtual reality describes a three-dimensional computer-generated environment which can be exploited and interacted by a person.

ii. Augmented Reality

   Augmented realty consists of patterns printed on physical objects, which can easily be tracked using computer vision virtually.

iii. Robotics

   Robotic applications are typically situated within the domain of space exploration, military-based research projects and smart homes to facilitate automation.

iv. Desktop, Tablet and Gaming Applications

   In desktop computing applications, gestures can provide an alternative interaction to the mouse and keyboard. Instances involve annotating and editing documents using pen-based gestures. Gestures can be used on in playing of video games with the help of a third-party hardware like the Microsoft Kinect.

v. Home Automation

   Home automation gives you access to control devices in your home from a mobile device anywhere in the world. The term may be used for isolated programmable devices, like thermostats and sprinkler systems

vi. Privacy – Unlocking Phones and Personal Computers

Users are able to define a pattern or formulate a specific movement of their physical body to be used while unlocking their phones and personal computers.

vii. Defense – Military

Military air marshals use hand and body gestures to direct flight operations aboard aircraft carriers.

viii. Automotive Industry

Driving without handling steering wheel and answering calls and other activities without vehicle movement being interfered with.

1.1.7: User Behaviors with Gesture-based interaction

Gesture-based interaction enable a natural and intuitive method for human-computer interactions for a myriad of computing domains, tasks, and applications. The maturing of sensing technology over the past few decades has brought gesture-based interaction accessible to everyone. The prevalence of gesture-based interaction has changed how people interact with computing systems in various domains, e.g., gaming, education, e-commerce, and healthcare.

Human gestures constitute a space of motion expressed by the body, face, and/or hands. Among a variety of gestures, hand gesture is the most expressive and the most frequently used. Gestures have been used as an alternative form to communicate with computers in an easy way. This kind of human-machine interfaces would allow a user to control a wide variety of devices through hand gestures. Glove-based gesture interfaces require the user to wear a cumbersome device, and generally carry a load of cables that connect the device to a computer(Alsheakhali et al., 2011).

Many virtual reality and augmented reality devices such as the Oculus Rift and Microsoft HoloLens, have incorporated gesture and motion tracking technology to facilitate a more immersive experience by allowing users to interact with the virtual environment using touch gestures, mid-air hand gestures, and body movements. The gesture movements are

interactive, image displays enhanced by special processing and by non-visual display modalities, such as auditory and haptic, to convince users that they are immersed in a synthetic space.

Gesture-based interaction changes how users interact with the digital environment. For example, consumers could directly touch and rotate the product presented on a touchscreen instead of clicking and dragging a mouse. Studies by (Alsheakhali et al., 2011) have demonstrated that gesture interaction can help to reduce risk of heart disease, diabetes and even early mortality, decrease risk of arthritis, frozen shoulder and cervical spondylosis, which cause chronic pain and also to reduce insomnia, anxiety and depression.

The ubiquity of common web cameras, coupled with the capabilities of modern processors, is quickly making gesture-based interaction a realistic input modality for a variety of computing devices and applications. For example, gesture-based input has been demonstrated for robotic control video game consoles and control of other home appliances. Immersive virtual reality presents an artificial environment that replaces users' real-world surroundings convincingly enough that they are able to suspend disbelief and fully engage with the created environment. Impressiveness is an important element of virtual reality application.

1.1.8: Fatigue in Gesture Computer Interaction – "*Gorilla arm"*

The use of gestures as a way to interact with computer systems has shown promise as a natural way to interact and manipulate digital information. However, users performing mid-air gestures for even moderate periods of time experience arm fatigue and discomfort, earning its name of the gorilla arm syndrome(Zhenhui et al., 2017). The gorilla arm syndrome originally arose with the advent of touchscreens used in a vertical orientation, which forces the user to extend their arms without support. When this is done for any task longer than a few minutes for instance with an ATM, it causes arm fatigue and a feeling of heaviness in the arms.

Fatigue levels are determined as the amount of time a person can maintain an isometric (static) muscle contraction. This can be measured by the heart's response to increase blood flow to transport oxygen to the muscle fibers (cells). An isometric muscle contraction causes an impairment of blood flow due to the increase in intramuscular pressure**.** Professional speakers and professors that must speak to audiences for 1-3 hours at a time are performing large quantities of unsupported hand gestures. How are they doing this without extreme fatigue in their arms and shoulders after performing gestures for hours?

Public speakers conduct most of their hand gestures in the comfortable zone with supported and mid-air gestures that are periodically relaxing their arms by extending them straight down, in a pocket, or taking advantage of a supporting structure like a podium. This periodic relaxation of the arms allows improved blood flow and therefore oxygen to the previously contracted muscles, which avoids switching to a glycolysis process that causes extreme arm fatigue as shown in the following figure 1.2.

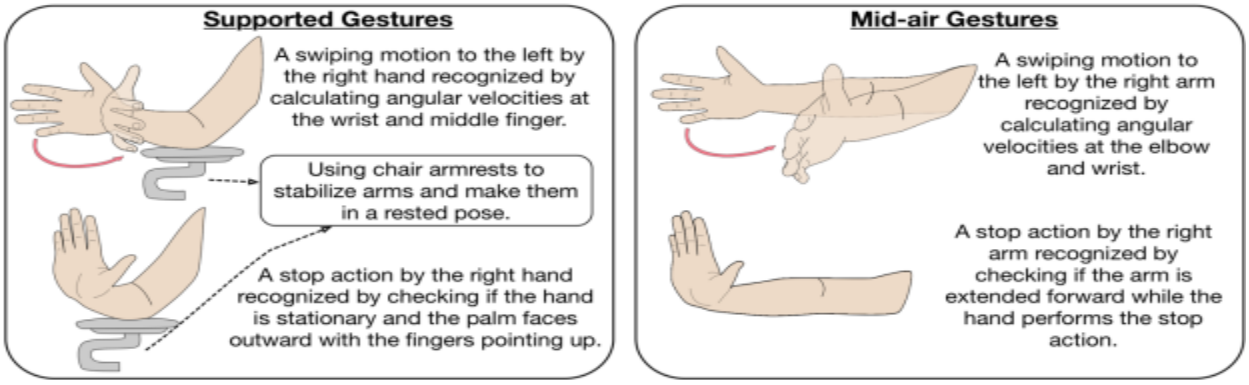


Figure 1.2: Supporting and Mid-air Gestures

1.1.9: What is Design Thinking

Design thinking is a user-centered approach to solving problems which draws on both the creative and rational thinking of multi-disciplinary teams to develop better solutions to various problems, including complex problems. Creativity is infused within the decision-

making process and provides constructive simple solutions to complex problems. All along, it is hard work for the developers but in the end ,the product must be sustainable (*In Design Wicked Problems Thinking*, 2008).

Design thinking is an essential tool for simplifying and humanizing. Every established company that has moved from products to services, from hardware to software, or from physical to digital products focuses on new user experiences ("Design Thinking for Problem Solving and Strategic Innovation," 2015). Organizations can upgrade with no constraints effectively when using gesture interactions as design thinking has some organized principles that guide a simple intuitive solution for the user's needs.

Design thinking focuses on three main elements of a solution: people, technology and business. All of these aspects revolve around the customer or consumer. The consumers are the key drivers of future technology consistency (Young & Consultant, 2010). Any technology solution, whether delivered to a small or larger group of people must create intrinsic value and address specific consumer needs. If the technology fails to create value to consumers, it will fail.

Design thinking makes the consumer the main focal point, constantly applying the values diversity and ambiguity as well as recognizing the importance of multi-disciplinary teams. The end point is the consumer for any product, whether technical or non-technical (Kimbell, 2012). It takes many iterations to create true innovation. Operating with the mindset of getting something in front of a consumer as early as possible and then keep going back after each refinement will help ensure the innovation stays human-centered and can progress quickly. Extreme clarity on objectives and action to be taken; Precise problem definition is always the starting point for applying design thinking and the same.

Design thinking is a discipline that uses the designer's sensibility and methods to match people's needs with what is technologically feasible and what a viable business strategy can convert into customer value and market opportunity. Design thinking includes at least two key viewpoints that of the designer and that of the decision makers; the study of the

practices of working designers and the other meaning refers to the human-centered ‘open’ problem solving process decision makers use to solve real world ‘wicked’ problems. Through this process, sometimes teams will discover that new consumer research is actually not the right next step.

1.2: Statement of the Problem

High quality gesture interaction hardware devices have become increasingly available at successively lower cost. However, companion gestural UIs have gained little traction despite their strong value proposition as they incorporate a lot of users physically chunk command and its operands into a single action. Gesture human computer interactions are more user-friendly and make it quick for manipulations to provide a platform for enjoyable computing.

User’s experience challenges when engaging and implementing gesture interactions, that is: complex coordination of all gestures regularly, mastering approachability to different gestures, muscle fatigue and a lack of supporting data input that is seamless with gesture interaction. ‘Unlike touch input, users usually have little experience with gesture interaction. Technologists frequently claim that gestural input makes computing more “natural” by enabling communication with a computer the same way we also communicate with one another’ (Angelini, 2017).

Gesture human computer interactions are meant to make end-user tasks simpler to use any computing device, unfortunately end users often struggle to execute gestures correctly. To compensate this, users require an efficient design thinking framework that supports them during the execution of a gesture. Human gesture interactions should be enhanced to suit users. Applied design thinking methodology is a user-centered approach which draws on both the creative and rational thinking of multi-disciplinary teams to develop better solution to complex problems.

1.3: Justification

Gesture-based human computer interactions are constantly being not taken into account by most end users due to complexity in the way they are designed and applied. End users stop using gesture-based human computer integration after one or two attempts due to complicated gesture input and output sets. There needs to be a way in which gesture based human computer interactions input and output sets are structured and implemented to enhance systems performances accompanied with gestures as gesture interactions input sets that are faster and present better user experience than the traditional physical input sets like mouse and keyboard.

Design thinking approach is all about embedding disclosure information in a familiar user interaction. It is about users encountering relevant gesture details only as needed, after they have formed a mental goal, searched for, and found an appropriate gesture behavior. Every established company that has moved from products to services, from hardware to software, or from physical to digital products focuses on new user experiences. A design thinking framework could provide knowledge about appropriate usage scenarios and develop a set of common gesture input sets for seamless end user's implementation which can enhance gesture based human interactions.

1.4: Research Objectives

1.4.1: Broad objective

The main objective of this research was to develop a framework that can promote a methodological approach to designing gesture interactions.

1.4.2: Specific objectives

The specific objectives of this research were to:

i. Analyze a set of high-level characteristics which can be compared and contrasted in gesture interaction.

ii. Develop a framework that supports design thinking activities towards promoting the use of gestures interactions.
iii. Evaluate the design thinking framework for gesture human computer interaction.

1.4.3: Research Questions

The following are questions that guided the research into achieving all its objectives:

i. How can a set of high-level characteristics be compared and contrasted in gesture interaction?
ii. Can a design thinking framework support and promote the use of gestures interactions?
iii. How can a design thinking framework for gesture human computer interaction evaluated within different tests and experiments?

1.5: Conceptual Framework

The interoperability of activities accounts for interpretations that influenced the overall research objectives and resulting questions which are fundamental for the whole study. The final developed design thinking framework should be user-centered to improve gesture computing experiences and innovation within human computer interaction with gestures.

The conceptual framework diagram for the entire research is presented in the figure 1.3 below.

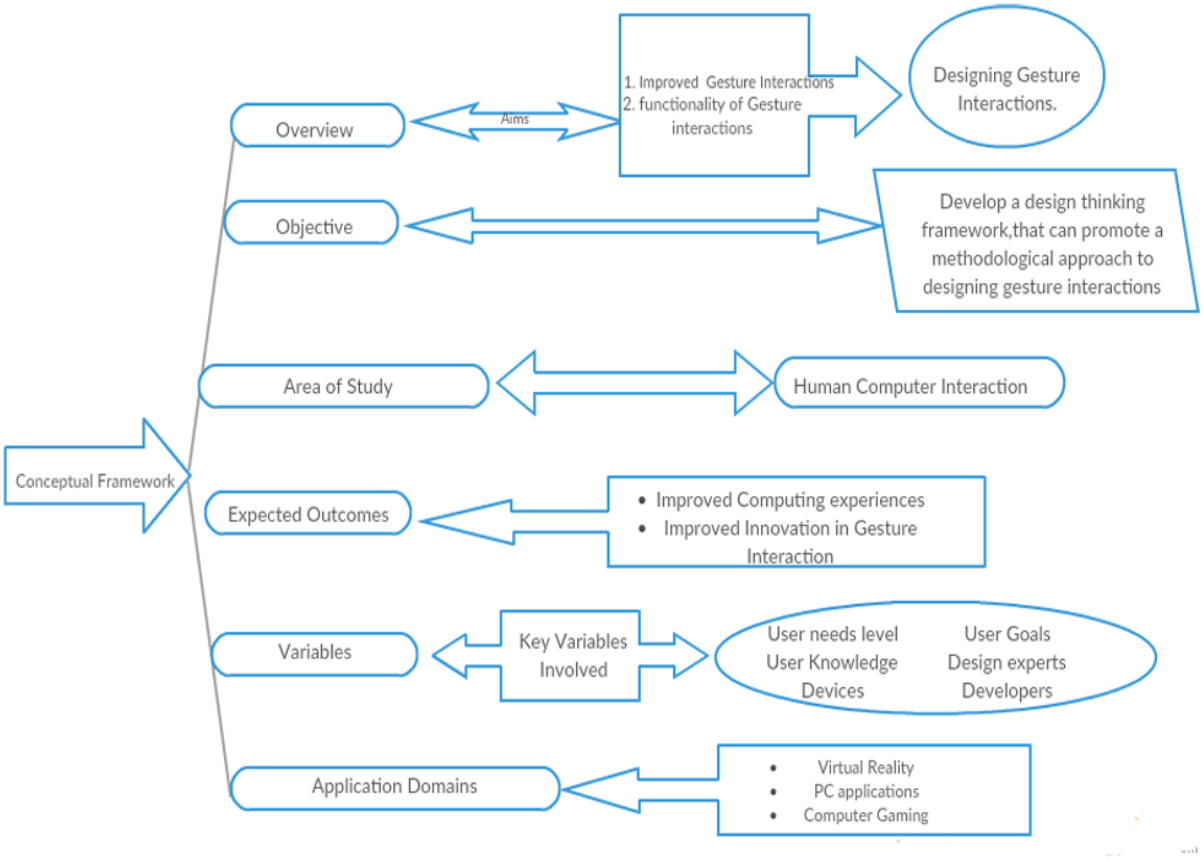


Figure 1.3: Conceptual framework

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

Human Computer Interface (HCI) was previously known as the man-machine studies or man-machine interaction. Human computer interaction deals with the design, execution and assessment of computer systems and related phenomenon that are for human use.

Usability has three components, that is: effectiveness, efficiency and satisfaction, using which, users accomplish their goals in particular environments. The early involvement of users has the potential for preventing serious mistakes when designing. Indeed, it compels designers to think in terms of utility and usability leading to completeness of system functionality, repair effort saving, as well as user satisfaction.

Gesture-based UI refers to using specific physical gestures in order to operate an interface. Take your smartphone for instance. Users can already interact with your phone without using the keypad by swiping, tapping, pinching, and scrolling. With some embedded GUIs, you can simply tilt or shake the device to engage with it using built-in accelerometers, gyroscopes, or magnetic sensors. Some products on the market today also support advanced camera and sensor technology that pick up facial expressions and eye movements to scroll, click, and interact (Kekong et al., 2021).

The major objective of gesture-based human interface is designing user-friendly interfaces or interactions. The other objectives of gesture-based user interface are:

i. Creating usable software-enabled products and user-interfaces.
ii. Enhancing the usability of existing products.
iii. Identify problems and tasks (such as in the workplace) that can be addressed with technological products.

iv. Saving costs that are incorporated earlier during user involvement

HCI has an associated design discipline, sometimes called *Interaction Design or User-Centered Design,* focuses on how to design computer technology so that it is as easy and pleasant to use as possible (Babu et al., 2015). In particular, gesture usability research recommend that system applications should use subtle visual feedbacks such as animations to help the user visualize the results of their actions, communicate the status, and enhance the sense of direct manipulation.

2.1.1 Theoretical Literature Review

Human computer gesture interaction input tasks could range from navigations, visual display and audio. Different users' approach gesture interactions tasks based on different levels of their knowledge, experience with computers and how easy gesture interactions tasks changed. Some users follow the entire structure, others do not. Most users revert back to using physical input or quit using the system gestural-based user interface does not have that alternative input sets.

Typical gesture interactions are evidently being adapted to improve computing and enhance different approaches for end users' solutions. From normal computing applications, gaming, health and educational practices. A study by (Dalka & Czyzewski, 2010) suggested that some users could successfully skip navigation of visual gestures in their course to navigate to audio gestures in an application of an online music web application to quickly get what they want.

Concepts from cognitive science and psychology have contributed to the field of human gestures, with phenomena on how they are conceived and provided and in-depth on how the gestures are classified. Classification of human gestures could be based on auctioning of activities, language that is being used, the way of execution and the expected outcome of the end-needs.

The cognitive science of gesture and gestural interaction aims to ensure a simple end means for a user to enjoy and interact well irrespective of different domain of their computing applications to a satisfactory computing experience. As observed by (Gope, 2011) , physical structures enable interactions that are very satisfactory for users, for instance the way normal users interact with their non-computing devices. There could be an easy way for end-users to efficiently use gesture interactions depicting their normal day to day interactions.

Gestures can be complex, involving multiple fingers, taps, and movement in all sorts of ways: up, down, left, right, circular, tapping, with one, two, three, or four fingers, add full body motion, speech, eye gaze, posture, etc. to the range of gesture input set tasks being sensed and it becomes a daunting challenge for gestural end-users to learn the required input actions and then to remember them (Billinghurst, 2015).

There are different types of gesture interaction design approaches and are classified into two major categories, that is model and fluid gesture design which are further subdivided into categories of other different usage gestures approaches based on single specific instance of a particular kind of human intents reaction to physical gestural motion.

2.2: Approaches to Gesture Interaction Design

The approach of gesture interaction design is divided into two mutually interdependent fields of inquiry: the model and fluid gesture design and are largely based on the disciplinary backgrounds from different previous research. Modern gestural academic investigations stem from (David. Efron, 2012) which laid out the first clear argument against the longstanding belief that speech and gesture are two fundamentally separate (and indeed, hierarchically valued) modes of relaying information that lead to conclusion that gestures share a speech computational input sets thereby both speech and gesture becoming parts of the same psychological structure.

Gestures in natural interaction consist of movements of hands, face or other body parts that are used for communication between people, replacing or enhancing speech. Computer gesture interfaces emulate such kind of communication, recognizing a set of gestures and exploiting them with a little speech as input for computing devices. Modern tracking and sensing technologies have been employed in order to recognize gestures input sets. Gestures interactions' have been adopted in a number of interactive systems.

Physical human actions are highlighted as an important resource for the users while moving gesture objects in a typical action that can be carried out by the users during input. Typical actions with human gesture actions are performed to communicate with hardware input, in particular, to emphasize particular properties of the human actions. A description of specific gesture input sets is a particular way that hardware sensors manipulate and understand human physical gesture actions and convert them as gesture input sets. Among the examples are reported systems where the system's hardware; manipulates human head to visualize brain sections and systems where the manipulation of a gesture input sets can be generated.

Touchless input is a massive competitor to gestural interactions and other the classical GUIs. Gesture Object Interfaces, accounts for the users performing gesture while enabling hardware sensors to respond to the GUIs, where the gestures are manipulated to interact with the information. Gesture interaction differs from touchless input recognition; where touch interaction constantly use the hand (direct control) for the user to control directly the digital information, and from gesture interfaces, while in touchless input the information is directly controlled through the manipulation of generated user physical control(indirect control) (Trigueiros, 2013).

The following presents the state of the art for the two categorized gesture interaction design approaches, that is the model and fluid gesture interaction applicable in smart homes, controlling the infotainment (human physical input sets materials intended both to entertain and inform) system used for interpersonal and emotional communication.

2.2.1: Model-Gesture Design

Model gesture design interfaces emulate communication by recognizing a set of gestures an exploiting them as input for computers. Model gesture interaction use tracking and sensing technologies to recognize gesture input sets from physical actions by the end-user based on gesture segmentation methods that are commonly inferred based on color filtering and are seriously affected by the appearance of skin color-like of human physicality and by lighting conditions to enhance accuracy of gesture input sets from the user (Muser, 2015).

More recent improvements to model gesture interaction design is depth sensors with the availability of synchronized RGB information where (N. Sharma et al., 2021) details how model gesture design combine color and depth information to obtain a satisfactory physical model segmentation for gesture input sets. A series of different filters is applied to remove all the parts of the image that do not correspond to the physical human skin regions. The second step classifies the different blobs into different human body parts, discerning between faces and arms. These color units enhance gesture input recognition by computer hardware.

Model gesture interaction design requires interaction with a system approach that is definitive of a multi-device user interface through some levels of abstraction that is the concepts and tasks level, the abstract user interface, the concrete user interface, and the final user interface. The concepts and tasks level contain the description of the concepts managed by the application together with the tasks that should be supported. The abstract user interface contains a user's description independent with respect to the device and the interaction modality. The concrete user interface contains a user interface description abstract with respect to the technology used for the implementation. The final user interface contains the final implementation of the user gesture input sets expressed in source code for the system's hardware to pick final gesture input set.

A model gesture interaction meta-design allows gesture interaction design address broad application domains (such as telecommunications, avionics, manufacturing, and robotics engineering) and are the cornerstone of gesture interaction input activities. Relative to system infrastructure and middleware integration frameworks, model gesture interactions are expensive to develop and purchase. However model gesture design provides a substantial return on sustainable gestures development since they support the development hardware inputs for user applications and products directly.

2.2.2: Fluid Gesture Design

Fluid gesture design systems run autonomously, mostly automated performing functions in response to user inputs. Many fluid gestures designed machines and devices are especially convenient because their electronic circuits sense and respond to human gestures. Automated entry/exit doors, for example, open by themselves when people walk up to them. Such basic non-contact human-machine interfaces (HMIs) do not change end-user's behavior, but come with limitations; fluid gesture design solutions detect only the presence of a body or hand, so the electronics could sometime respond inappropriately, For instance, a door might open unexpectedly when a pedestrian walking by and pauses momentarily near the entrance leading to a bad user experience (Palacios et al., 2013).

Fluid gesture design however; make erroneous system operations much less likely; In a typical application, for example, a person standing in front of a door could just hold out a hand and move it from left to right to open the door, or move it from right to left to close it. Device and equipment manufacturers increasingly want to incorporate fluid gesture-recognition approach into new embedded system products.

Fluid gesture models can recognize sequential data using a probabilistic approach. Series of observations are modelled using a finite number of states, whose transitions are defined by transition probabilities. Each state emits observations based on a probability distribution function. Generally, fluid gesture interactions parameters are set through training procedures using a large database statistically representative of all possible variations.

Fluid gesture approach are limited by interactive procedure between designing gestures and receiving feedback on the gestural interface behavior, hence fluid gesture interaction is designed to allow users to rapidly define and test gestures as required based on specific hardware requirement. This is why the user early learning procedure of input needs to be as quick and simple as possible. Fluid gesture interaction uses a hybrid approach between probabilistic gestural-based user interfaces.

Fluid gesture design recognizes gestures only after they have been entirely completed as happens in classic gesture recognition systems, also exploiting the full potential of gestural interaction by tracking gestures continuously and synchronously. Fluid gesture design allows users to both control the target application moment-to-moment and also receive immediate and synchronous feedback about system recognition states. Furthermore, fluid gestures are the pre-defined with gestural input sets which allows users to design their own gestures so making interaction more natural and also allows the system applications to be tailored by users' specific system hardware.

2.3: Related Frameworks for Gesture Interactions

The popularity of gesture-based interaction is built on the advancement in sensing, motion tracking, and computer vision technology. Extensive research effort has been devoted to improving gesture recognition and tracking performance (Rautaray and Agrawal 2015). Despite the technological challenges, it is well recognized that many challenges users are facing when interacting with gesture-based devices are about usability, that is how to ensure that the capabilities of the technology are well matched to the needs and capabilities of the people who use them.

In spite of available designed gesture interaction frameworks from user studies, researchers and practitioners have developed different embodied approaches to guide intuitive gesture-based interaction design. (Dourish 2004) used the term embodied interaction to describe an approach that placed an emphasis on understanding and incorporating our relationship with the physical world around us into the design of interactive systems. Embodied interaction is an approach to understanding human–computer interaction that seeks to investigate and support the complex interplay of the mind, body, and the environment during interaction.

Gesture-based interaction input constitutes visual, audio, and tactile feedbacks (Blackler and Popovic 2015). Unlike interacting with real world objects, users need to rely on the feedback triggered by their gestures to understand the virtual environment. Therefore, a stream of literature investigates how multisensory feedback could facilitate usability of gesture-based interaction. First, feedback can inform users on how and where to gesture. When exposed to continuous visual feedback. Feedback can also provide a sense of control by informing users whether an action is indeed working appropriately.

All gesture-based interaction have advantages including naturalness and expressiveness, learnability, freedom and control, and the ability to leverage existing dexterous skills and elicit positive emotions from users. The following related literature and frameworks designed for gesture interaction developed to enhance computing with the use of gesture interaction inputs. These approaches address directly gesture interaction. with human physical actions in a wider context, or for specific application domains or gesture types.

2.3.1 Sonification of Affordances for Gesture-Based Interfaces

Sonification (process of translating data into sound) can play a significant role in facilitating continuous, gesture-based input in closed loop human computer interaction, where it offers the potential to improve the experience of users, making systems easier to use by rendering their inferences more transparent. The interactive framework in figure 2.1 described by (Visell & Cooperstock, 2007) here provides a number of gestural affordances which may not be apparent to the user through a visual display or other cues.

This framework approach combines machine learning techniques for understanding user's gestures, with a method for auditory displays of salient features of the underlying inference process in real time thereby providing user with desired gesture inference sets shown in the following figure 2.1

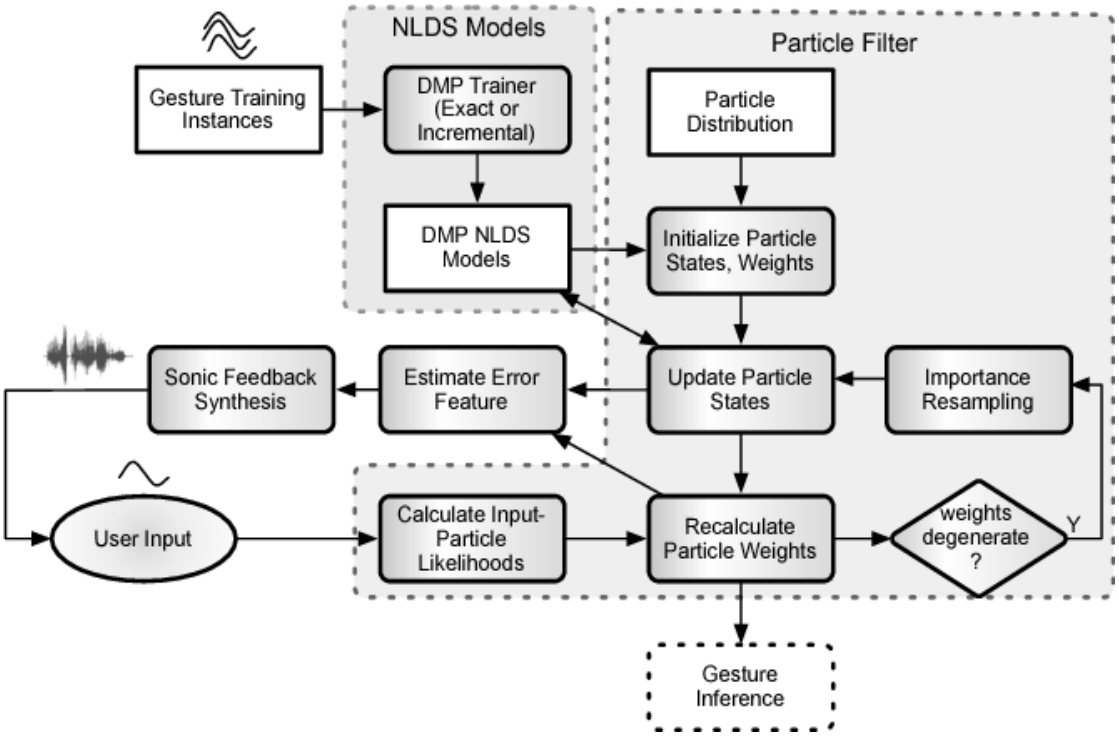


Figure 2.1: Sonification of Affordances for Gesture-Based Interfaces

Gesture-based input from the user is acquired by the input device's sensors as an N-dimensional temporal trajectory. A set of particles, given by weighted states of the nonlinear dynamical systems, is evolved in tandem with the gestural input. They maintain a probability distribution that tracks the system's belief as to the correspondence between the user's input and learned models. The error signals and weights that result is then used to derive parameters for a sonification.

The role for sonification in this system is to convey information in order that users may better guide their actions relative to the inferences made by the system. The approach

adopted is that of parameter mapping which extracts meaningful values from the system state in order to drive the parameters of a sound synthesizer. The current situation, in which the output of an interactive system is fed to a sound synthesis engine, is notably analogous to the mapping problem encountered in the control of sound synthesis with digital musical instruments.

The new architecture for the interactive sonification of gestural affordances, based on a combination of modern tools for gesture tracking, modeling, and new proposals for the derivation of sonification parameters aids to a new approach. The sonification of gesture approaches to the gesture interaction problem should return to the setting of closed loop control with sonic feedback in order to comprise a sufficiently rich body of gesture examples from which users can continue to enjoy gesture interaction with systems

A key problem with this framework approaches to sonification is the determination of main gestural-based input parameters for display, which is analogous to the control mapping for gesture interaction input. The state of each particle in the sample-based displays for the control systems they describe are mapped onto the playback parameters for sound in a granular synthesis scheme. Provided a sufficient density of particles, it is plausible that the sonic texture that results evidence the underlying probability density that the particles sample, rather than the particular sample set, in the same manner as described above, and, in any event, what doesn't survive might be perceived as an unbiased noise.

2.2.2 A Framework for User-Defined Gestures for Surface Computing

This framework by (Wobbrock et al., 2009) presents effects of gestures to users and elicits the causes meant to invoke them by using a think-aloud protocol and video analysis, to obtain rich qualitative data that illuminates users' mental models while they are using surface devices like tablets. The effects obtain quantitative measures regarding gesture timing, activity, and preferences resulting in a detailed picture of user-defined gestures and the mental models and performance that accompany them.

Data input was from non-technical people aged 18 to 43 without prior experience using touch screens (e.g., the Apple iPhone), expecting that they would behave with and reason about interactive tabletops differently than designers and system builders. Each command's conceptual complexity was rated by the 3 authors (1=simple, 5=complex). During the study, each command was presented with an animation and recorded verbal description from 27 commands for which as shown in the following table 2.1

Table 2.1: Variants for a User-Defined Gestures for Surface Computing - The 27 commands for which participants chose gestures. Each command's conceptual complexity was rated by the 3 authors (1=simple, 5=complex). During the study, each command was presented with an animation and recorded verbal description

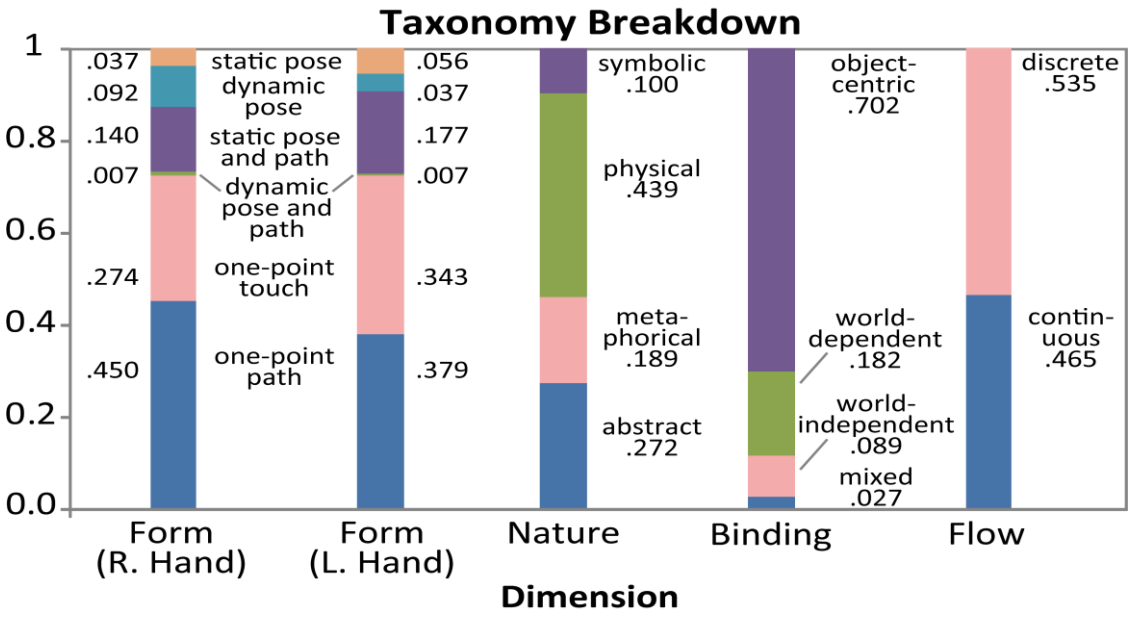
REFERENTS	*Mean*	*SD*	REFERENTS	*Mean*	*SD*
1. Move a little	1.00	0.00	15. Previous	3.00	0.00
2. Move a lot	1.00	0.00	16. Next	3.00	0.00
3. Select single	1.00	0.00	17. Insert	3.33	0.58
4. Rotate	1.33	0.58	18. Maximize	3.33	0.58
5. Shrink	1.33	0.58	19. Paste	3.33	1.15
6. Delete	1.33	0.58	20. Minimize	3.67	0.58
7. Enlarge	1.33	0.58	21. Cut	3.67	0.58
8. Pan	1.67	0.58	22. Accept	4.00	1.00
9. Close	2.00	0.00	23. Reject	4.00	1.00
10. Zoom in	2.00	0.00	24. Menu access	4.33	0.58
11. Zoom out	2.00	0.00	25. Help	4.33	0.58
12. Select group	2.33	0.58	26. Task switch	4.67	0.58
13. Open	2.33	0.58	27. Undo	5.00	0.00
14. Duplicate	2.67	1.53	**MEAN**	**2.70**	**0.47**

After all, 20 participants had provided gestures for each referent for one and two hands, the gestures were grouped within each referent such that each group held identical gestures. Group size was then used to compute an agreement score that reflected, in a single number, the degree of consensus among participants.

The creation of a user-defined gesture sets for surface computing is based on observed user behavior and joins gestures to commands. Referents use reversible gestures, and the same gestures are reused for similar operations. For instance, enlarge, which can be accomplished with four distinct gestures, is performed on an object, but the same four gestures can be used for zoom in if performed on the background, or for open if performed on a container (e.g., a folder). Flexibility exists insofar as the number of fingers rarely matters and the fingers, palms, or edges of the hands can often be used interchangeably.

The form dimension is separated by hands for all 2-hand gestures. The percentage of gestures in each taxonomy categories from top to bottom are listed in the same order as they appear in the following table 2.2.

Table 2.2: taxonomy categories for user defined surface computing - Percentage of gestures in each taxonomy category. From top to bottom, the categories are listed in the same order as they appear in Table 2. The form dimension is separated by hands for all 2-hand gestures. (All participants were right-handed)



There are drawbacks to this framework approach as users could not change previous gestures after moving on to subsequent ones; perhaps users would have performed differently if they first saw all referents, and then picked gestures in an order of their choosing. Application context could also impact users' choice of gestures, as could the larger contexts of organization and culture. The user-defined gesture set cannot be implemented with a vision-based gesture recognizer so that system performance and recognition rates can be measured.

2.2.3: A Framework for User-Defined Motion Gestures for Mobile Interaction

Modern smartphones contain sophisticated sensors to monitor three-dimensional movement of the device. These sensors permit devices to recognize motion gestures, deliberate movements of the device by end-users to invoke commands. However, little is known about best-practices in motion gesture design for the mobile computing paradigm.

To address this issue, this framework presents the results of research that elicits end-user motion gestures to invoke commands on a smartphone device (Tse et al., 2006).

This framework demonstrates that consensus exists among the participants on parameters of movement and on mappings of motion gestures onto commands to develop a taxonomy for motion gestures and to specify inspired motion gesture set for end-users. Figure 2.2 illustrates the agreement for the gesture set developed by participants. As shown by their agreement scores, there was not a consensus on a motion gesture for switching to next application, switching to previous application, and act on selection tasks.

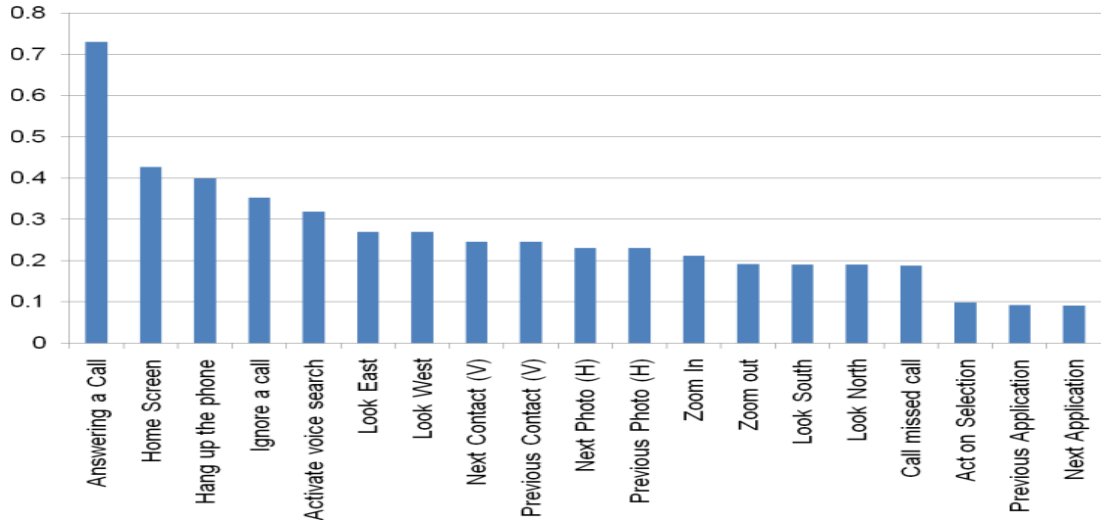


Figure 2.2: Agreement Scores for User-Defined Gestures task assorted in ascending order

The problem with the framework is that it requires user thoroughly view the screen while interacting with gestural-based user interface. Since the tasks selected in the framework required the user to interact with content on the screen after completing each gesture navigation.

2.4: Design Thinking

2.4.1: What is Design Thinking

Design thinking is a means of solving complex problems that focuses on the perspective of end users to better determine application requirements. Design thinking principles have been utilized by some of the world's most influential technology corporations such as SAP, IBM, Apple, Uber, Airbnb, and Capital One as a means of developing better products and services. The concepts of innovation and empathy are a reoccurring pattern in design thinking as a development methodology. In the traditional project management and system development methodologies, whether waterfall or agile, customer interaction and participation is mostly limited to a specific time set aside to determine user requirements (T. Brown 2008)

The origins of design thinking date back to the 1960s the so-called design science decade and are rooted in the early works of design methodologists. Thus, while natural sciences dealt with the analysis of existing reality, the science of design dealt with "the transformation of existing conditions into preferred ones". Later conceptualizations of design challenged the positivist doctrine characterizing the "design science" movement and embraced a more constructivist stance to design as a practice.

The designers' attitude to solve such problems uniquely from normal thinking as well-defined "puzzles" to solve, design problems were described as ill defined, ill structured. Furthermore, they were described as open ended and, initially, as highly ambiguous and presenting multiple plausible solutions (Goldschmidt,2007). Therefore, the designer's task was identified as producing an appropriate solution by "organizing complexity and finding

clarity in chaos" through a process of patterned synthesis of aesthetic, cultural, and technology trends and consumer and business needs (Kolko, 2010). Abductive reasoning was said to rest on designers' efforts to find plausible solutions, where design thinking was realized.

Design thinking builds on the process of interacting with the customer from the start till the end of a customer's needs and environment. As the name implies, design thinking is a problem-solving framework and not an exclusive project execution framework such as waterfall and agile. Design thinking starts by defining the problem and then developing a solution with a focus on the customer or user of the final product. As you focus on understanding the customer's problem, you can then create prototype solution. This prototype is then tested allowing you to continue to learn and improve upon your solution (Steinke et al., 2017).

2.4.1: Design Thinking in Practice

There is a shift under way in large organizations, one that puts design much closer to the center of the enterprise. But the shift isn't about aesthetics. It's about applying the principles of design to the way people work (Essays, 2013).

This new approach is in large part a response to the increasing complexity of modern technology and modern business. That complexity takes many forms. Sometimes software is at the center of a product and needs to be integrated with hardware (itself a complex task) and made intuitive and simple from the user's point of view (another difficult challenge). In opposition to the traditional design process, design thinking focuses on upfront problem framing before solutions are explored. It is important that the framing be independent of the solutions. Design thinking is an iterative process and relies heavily on prototyping to build knowledge, test and validate concepts. Design thinking helps non-designers understand and deal with ambiguity though the use of a structured process.

A dozen of other types of complexity that businesses grapple with every day all have one thing in common: People need help making sense of them. Specifically, people need their interactions with technologies and other complex systems to be simple, intuitive, and pleasurable. The role gesture interaction plays in design thinking and the role it plays in design collaboration. have been primarily concerned with studies of verbal protocols and very little on addressing consistency concerns.

,(Kimbell, 2012) raises awareness of the role of gesture in designing, primarily through the research on gesture when designers communicate and collaborate, but also in the implications of research related to gesture and thought on the design of HCI devices. A set of principles collectively known as design thinking; empathy with users, a discipline of prototyping, and tolerance for failure chief among them; is the best tool we have for creating those kinds of interactions and developing a responsive, flexible organizational culture.

2.4.2: Principles of Design Thinking

Design thinking converts need into demand. It's a human-centered approach to problem-solving that helps people and organizations become more innovative with the following listed principles by (Plattner, 2009).

"

i. Start with needs**user needs not system needs* - Empathy
   Service design starts with identifying user needs. If you don't know what the user needs are, you won't build the right thing.
ii. Do less
   Find a way of doing something that works and make it reusable and shareable instead of reinventing the wheel every time.
iii. Design with data
   Learn from real world behavior by looking at how existing services are used. Let data drive decision-making, not hunches or guesswork.

iv. Do the hard work to make it simple

Making something look simply is easy. Making something simple to use is much harder, especially when the underlying systems are complex.

v. Iterate. Then iterate again.

The best way to build good services is to start small and iterate wildly. Refinements based on feedback. It makes big failures unlikely and turns small failures into lessons.

vi. This is for everyone

Accessible design is good design. Everything built should be as inclusive, legible and readable as possible.

vii. Understand context

Designing for a screen, designing for people. Think hard about the context in which they're using our services.

viii. Be consistent, not uniform

Using the same design patterns wherever possible., as it helps people get familiar with your system.

ix. Make things open: it makes things better

We should share what we're doing whenever we can. With colleagues, with users, with the world. Share code, share designs, share ideas, share intentions, share failures.

2.4.3: Innovation with Design Thinking

The different models of design thinking including the IDEO model and IBM design thinking model share the target of achieving innovation through three main factors as shown in the following figure 2.3 below:

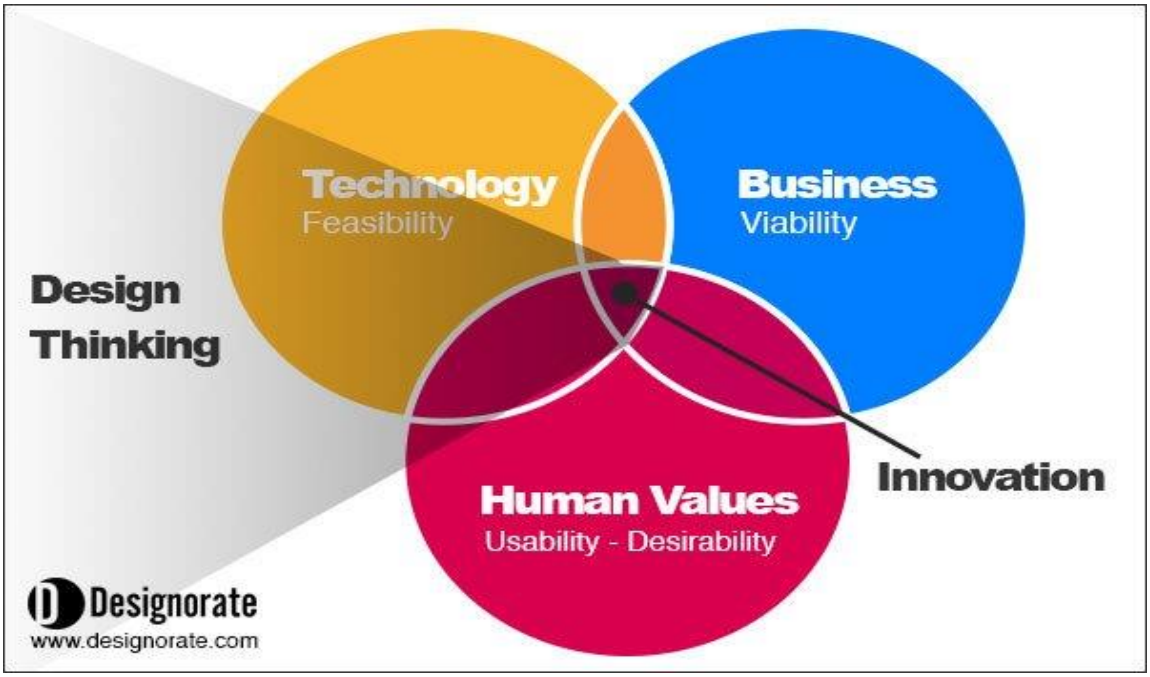


Figure 2.3: Design thinking model for innovation

Design thinking and innovation must factor into the following described attributes for longevity of desired solution as detailed by (Baert, 2015) ;

i. User Desirability. The product should satisfy the consumer's needs by solving everyday problems through a user-centered process. This can be achieved through a deep understanding of the user and through an empathic design process, which can only be achieved by putting ourselves in the shoes of our consumers.
ii. Market Viability. Successful products require an integrated marketing strategy that identifies the target segment and builds the product brand in accordance with this target segment. Tools such as the business model canvas can help our understanding of the project and create a business strategy.
iii. Technology Possibility. Technology provides state-of-art tools for designers to innovate and build products that meet today's needs. Technology should be adopted through the development process, including the prototyping stage where a visual presentation of the product is made to the team.

2.4.4: Characteristics of a Design Thinker in relation to Gesture-Based Human user Interfaces

In general, a good designer should be able to flexibly use different problem-solving strategies and choose the one that best meets the requirements of the situation. Regardless of the given problem, successful designers clarify requirements, actively search for information (i.e., critically check given requirements and question their own requirements), summarize information of the problem into requirements and partially prioritize them, and do not suppress first solution ideas.

Although the nature of design thinking and what makes one person a design thinker and another not remain elusive, a number of characteristics have been identified and can be useful in understanding how a design thinker thinks. The following is a summary of the design thinker characteristics that Owen (2007) described in relation to gesture-based human user interfaces.

i. Human- and environment-centered concern - Designers must continually consider how what is being created will respond to human needs. They should also consider environmental interests at a level with human interests as primary constraints for the design process.
ii. Designers work visually (i.e., depiction of ideas). - Designers should look at different/multiple solutions to a problem and keep the big picture of the problem in mind while focusing on its specifics.
iii. Designers should be able to verbally explain their creative process forcing invention where detail is lacking and expressing relationships not obvious visually (i.e., explanation should go hand in hand with the creative process).
iv. Designers search competing alternatives before moving to choose making or decision making. They try to find ways to come up with new configurations. This process leads to a solution that avoids decision and combines best possible choices.
v. Designers need to develop interpersonal skills that allow them to communicate across disciplines and work with other people.
vi. Systematic vision - Designers should treat problems as system problems with opportunities for systemic solutions involving different procedures and concepts to create a holistic solution

2.4.5: Design Thinking Competency Process Model

The competency operationalization of the design thinking construct helps drive the creation of appropriate activities that would allow for the collection of relevant evidence to inform

variables in the model. Skills associated with this variable include tinkering, creating, and testing ideas via diagrams. Testing, in turn, entails initial testing of the design idea, getting feedback, modifying the design, reevaluating it, and making a decision to accept or reject the modeled idea (Baert, 2015) . To assess competency levels relative to the iterate diagrams variable, we would have to put them in a situation in which those constituent skills could be employed, such as in a game or simulation.

The design thinking competency process model which diagnostically provides a strong foundation for evaluating the degree to which complex problems are approached with simple technological solution outlined by (D.School, 2013) is shown in the following figure 2.5

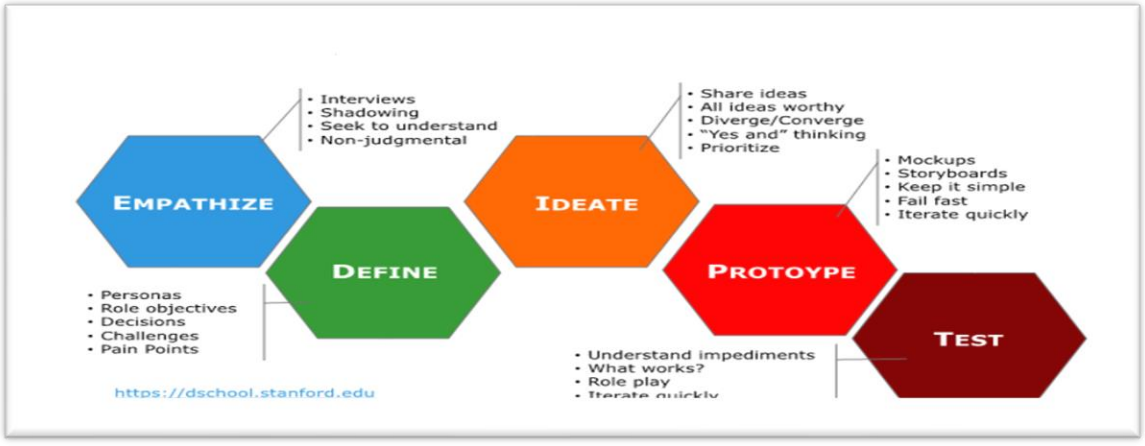


Figure 2.4: Design thinking Competency Model

2.5 Artificial Neural Network Models (ANNSs)

2.5.1 Introduction

Artificial neural networks (ANNs) are biologically inspired computer programs designed to simulate the way in which the human brain processes information. ANNs gather their knowledge by detecting the patterns and relationships in data and learn (or are trained) through experience, not from programming. An ANN is formed from hundreds of single units, artificial neurons or processing elements (PE), connected with coefficients (weights), which constitute the neural structure and are organized in layers. The power of neural computations comes from connecting neurons in a network (Kekong et al., 2021).

Each PE has weighted inputs, transfer function and one output. The behavior of a neural network is determined by the transfer functions of its neurons, by the learning rule, and by the architecture itself. The weights are the adjustable parameters and, in that sense, a neural network is a parameterized system. The weighed sum of the inputs constitutes the activation of the neuron. The activation signal is passed through transfer function to produce a single output of the neuron. Transfer function introduces non-linearity to the network. During training, the inter-unit connections are optimized until the error in predictions is minimized and the network reaches the specified level of accuracy (Singla & Patel, 2020).

Once the network is trained and tested it can be given new input information to predict the output. Many types of neural networks have been designed already and new ones are invented every week but all can be described by the transfer functions of their neurons, by the learning rule, and by the connection formula. ANNs require no knowledge of the data source but, since they often contain many weights that must be estimated, they require large training sets.

The artificial neuron is the building component of the ANN designed to simulate the function of the biological neuron. The arriving signals, called inputs, multiplied by the connection weights (adjusted) are first summed (combined) and then passed through a transfer function to produce the output for that neuron. A neural network is trained to map a set of input data by iterative adjustment of the weights (N. Sharma et al., 2021).

2.5.2 ANN Models and Learning Algorithms

There are many different types of ANNs, some of which are more popular than others. When neural networks are used for data analysis, it is important to distinguish between ANN models (the network's arrangement) and ANN algorithms (computations that eventually produce the net- work outputs).

Once a network has been structured for a particular application, that network is ready to be trained. There are two approaches to training, supervised and unsupervised. The most often used ANN is a fully connected, supervised network with backpropagation learning rule as shown in figure 2.5, is excellent at prediction and classification tasks. Another is the self-organizing map with unsupervised learning algorithm, which is excellent at finding relation- ships among complex sets of data shown in figure 2.6

The goal in supervised learning is to predict one or more target values from one or more input variables. Supervised learning is a form of regression that relies on example pairs of data: inputs and outputs of the training set. Supervised learning has fully inter- connected

neurons organized in layers, the input layer, the output layer, and the hidden layers between them. The input layer neurons receive data from a data file. The output neurons provide ANN's response to the input data. Hidden neurons communicate only with other neurons. They are part of the large internal pattern that deter- mines a solution to the problem (Morra et al., 2019).

The information that is passed from one processing element to another is contained within a set of weights. Some of the interconnections are strengthened and some are weakened, so that a neural network will output a more correct answer. The most commonly used learning algorithm is back propagation of error. The error in prediction is fed backwards through the network to adjust the weights and minimize the error, thus preventing same error from happening again. This process is continued with multiple training sets until the error is minimized across many sets. This results in the mapping of inputs to outputs via an abstract hidden layer.

2.5.3 ANN applications with Gesture Based Human Interaction

Expert systems are knowledge-based systems, an extension of conventional computing and are some- times called the fifth generation of computing. This knowledge base allows an expert to define the rules that simulate a process of thinking and provides a simple way to draw conclusions and solve problems by following a set of rules.

The idea of expert systems is that logical thinking can be modelled by compiling lists of logical propositions and performing logical transformations upon them. ANN provides a guide for prediction and decision making in environments involving uncertainty and vagueness as applied by the following different applications.

A gesture is known as the data obtained from a movement made with a combination of hardware sensor signals. (Lee-Cosio et al., 2012) presents the application of Artificial Neural Networks to recognise among gestures trajectory patterns in a Euclidean space. As shown in figure 2.5 below

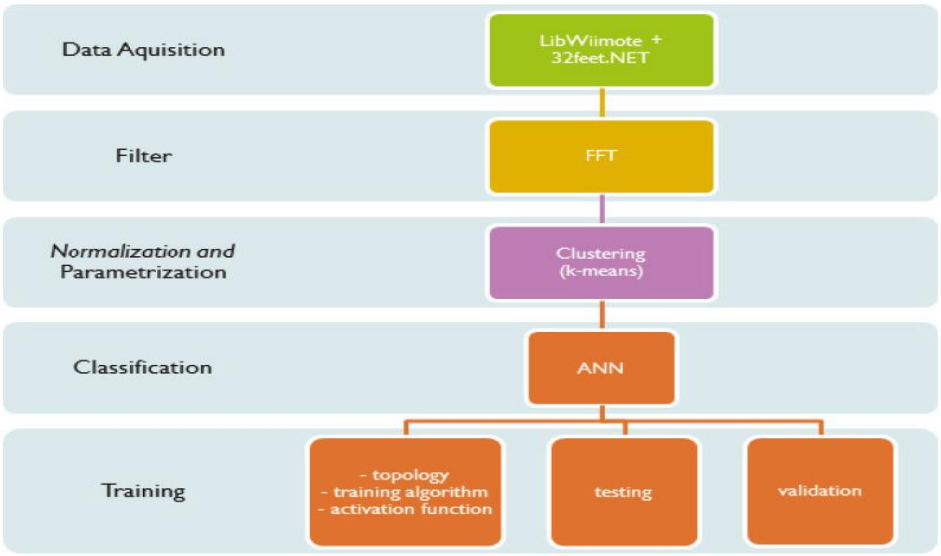


Figure 2.5: The application of Artificial Neural Networks to recognize among gestures trajectory patterns in a Euclidean space.

2.6: Gaps

Few frameworks address the development of gesture interaction system from a broad point of view, most focusing on specific gesture recognition techniques, dealing only with free-hand gestures or with multitouch gestures. The review offered the largest vision on enabling technologies for gesture recognition, distinguishing between perceptual systems and non-perceptual systems (those that require physical contact from the user). As a result of the gesture interaction limitations, researchers and designers have regrouped to come up with different gesture interaction frameworks and systems that enhance the push for gestures interaction development and consistency of use in the different application domains.

This study overhauls the common way gestures are designed within hardware systems rendering touch-less application graphics and status information using to give user desired process output with their gesture interaction input. The user-defined gesture input set should be implemented with a vision-based gesture recognizer so that system performance and recognition rates can be measured. A key problem in gesture interaction approaches is managing entire user journey in the determination of salient parameters for display and those not to, which is analogous to the control for the user.

User defined gestures for motions interactions maintained gestural extension while users are not looking at the screen and then loose feedback which seems undesirable for novices. Since the tasks selected in the frameworks required the users to interact with content on the screen after completing the gesture navigating from previous and next enlists without user awareness who is not in control and consistently interrupted. That is, a person cannot judge moments where they can stop, take over and/or fine-tune another person's actions

Furthermore, the review details challenges of working gesture interaction frameworks; with a key consistent challenge in all the frameworks being unexpectedly performance of a functionality and classification of different gesture styles. Eye gaze and consequential gesture communication helps people mutually understand when a required gestural event is about to happen, enabling cooperation vs conflicts with hardware system. A result of which the study is guided to establish a design thinking framework that can promote a methodological approach to designing gesture interactions.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Introduction

This chapter introduces experimental research design and its rationale to achieve research objectives. It justifies the choices for the techniques and methods used in experiments for gesture interaction based on design thinking. The methods used in the research experiments maximize on quality of output based on datasets input by addressing their potential limitations and domain applications.

Experimental research design is a study that strictly adheres to a scientific research design. It includes a hypothesis, a variable that can be manipulated by the researcher, and variables that can be measured, calculated and compared. Most importantly, experimental research is completed in a controlled environment (Boaz Cruz 2012). The controlled environment should have the maximum data requirements to achieve required research outputs.

The research methodology accounts for gathering the right experimental environment arrangement or card-indexing them, participation of experiment users when required, and also training in techniques for the collection of data appropriate to particular problems, in the use of required programming languages and the required controlled experimentation in recording data output evidence, sorting it out and interpreting it.

This chapter guides the research strategy and the empirical methods, and specific techniques to address the first and second objectives for the research. It also presents the research design and the methods used in the selection of software tools for simulation, how the simulation was executed. The experimental approach describes logic of development of the process used to generate theory that is procedural framework within which the research was conducted. It provides the principles for organizing, planning, designing, and conducting simulation.

Experimental approach research not only guides the selection of data gathering and analysis methods but also the choice of competing methods of theorizing test users involved in simulation experiment. Test users should achieve all designed variables for the datasets input in order to achieve quality output. Experimental research is guided by three major fundamentals: independent variables, dependent variables and the control mechanisms to ensure quality output of the variables.

3.2: Research Design

The main objective of this research was to implement a design thinking framework that can promote a methodological approach to designing gesture interactions. The design thinking framework is based on three main core entities which are: gesture output, analytics and feedback.

The following figure 3.1 shows the implemented design thinking framework developed using Google IO programming language.

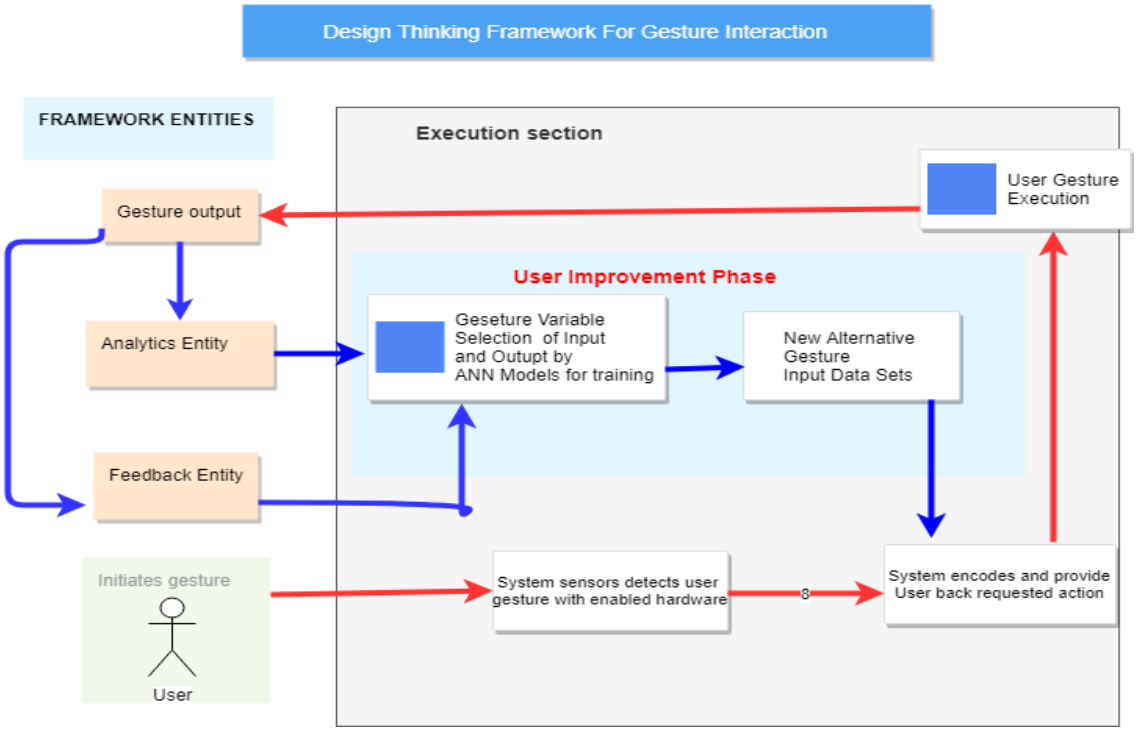


Figure 3.1: Design thinking Framework for Gesture-Based Human Computer Interaction

Figure 3.1 demonstrates how the design thinking framework improves gesture based human computer interaction process. The red path indicates path before framework is introduced; how gesture is captured from user with hardware sensors, system encodes gestures then a final gesture output is provided to the user. If a gesture is wrongly executed by user, the system does not provide an alternative gesture input, therefore failed gesture execution.

The implemented design thinking framework explicitly takes the sequence of sub-gestures for a given gesture set and uses it to learn ANNs model for subsequent gesture sets with different learning algorithms. This provides user greater flexibility in modeling gestures sets as feedback, analytic and output entities constantly iterating smaller sequence of actions for new and different gesture input sets.

Gesture output entity contains the final state of implementation where a report is updated into the feedback and analytics entities to recognize if a gesture output is completed or not. This allows defining gesture output sets based on user commands either using hands, legs or full body motion, which is optimized for the framework. Gesture output entity sends an identifier containing completed and incomplete gesture sets to the feedback and analytics entities.

The feedback and analytics entities define incomplete gesture input sets, and find the optimum parameters for ANN topology models. To do so, gesture input data from feedback, analytics and output entities is classified into two steps. First step is learning step (training step) in which a classifier is built to describe a predetermined set of new input data classes from the three entities. In second step the model which is built in first training stage is used for classification with a classifier accuracy of known data to new alternative gesture input data.

3.3 Simulation Environment

The required software tools for the simulation environment were gathered differently. The required hardware computing environment required specifications were a minimum of four GB RAM, a webcam camera, 500 gigabyte internal hard disk space and a CPU of 3.2 gigahertz clock speed. The following programming language were installed on the hardware computer.

Python3 is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python3 build the entire structure code developed for hand palm gesture interaction. Python3 made it to achieve high-level interaction nature of scientific libraries for a good platform of development.

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It efficiently provided symbolic math library used for neural network that helped in development of gesture interaction models for re-training hand palm images for alternative gesture input.

Tflearn is a deep learning python library featuring a higher-level API for TensorFlow. Tflearn made it possible to speed up hand palm gesture interaction execution and maintain compatibility between python programming language and TensorFlow.

OpenCV (cv2) is python3 library that enabled loading of hand palm gesture interface images from webcam of user input and python3 programming language.

NumPy is a library for the Python programming language, that facilitated the support for large, multi-dimensional arrays and matrices along with a large collection of high-level mathematical functions to operate on hand palm gesture interaction arrays.

Pillow (PIL) is a free library for the Python programming language that provided adding support for opening, manipulating, and saving many different images file formats for hand palm gesture interactions.

Imutils is a series of convenience functions to make basic image processing operations which facilitated the translation, rotation, resizing, skeletonization, and displaying hand palm gesture interaction images easier with Python3 programming language that was used for development.

3.3.1 Data sets and Model trainer

Custom image datasets of the hand gesture palm were generated from user gestural hand palm input. There were downloaded palm image of datasets around 10 imagery datasets from GitHub repositories to enhance retraining of models. A python library called Imutils enhanced the real time body gesture interaction with live gestural input via webcam. Imutils library has a series of convention functions that have in-built machine learning procedure that prompts opening the webcam to enable real-time interaction of gestures invoked by user.

Images were resized by rimage.py file so that they could feed into the Artificial Neural Network designed using TensorFlow which automated imagery of the hand palms real-time gesture interaction datasets against preconditioned variables (number of gestural input sets, error-rate, Average user-time and average execution tine-rate) being recorded in relation to accuracy of gesture execution and average time used.

3.3.2 Running the Simulation

A gesture predicator file is executed and a window named gesture interaction appears on screen of the user followed by another window named gesture input appearing on the same window screen. The next step prompts the user to start gesture input and output interaction in real-time. The user performing live gestural interaction with hand palm in three different states which involves either while standing, dancing, sited or jumping. The states account to compare the correct posture for gestural execution. A tracker.py file records variables being investigated including gesture execution accuracy, error rate of the different states and timestamp turnaround of gesture execution.

CHAPTER FOUR

MODELLING, ANALYSIS AND DISCUSSIONS

4.1 Introduction

After implementing the design thinking framework for gesture interaction, the immediate task was to check value and actualization of the framework in actual use. The purpose of modelling, analysis and discussion was to confirm that the framework improves end users in-demand need of human computer-based gesture interaction. This chapter discusses in detail the modeling design, analysis and results of the framework to an acceptable version that is of value.

4.1.1 Gesture Input

Test users were required to perform various hand palm gesture actions in front of user computer camera (preferable using a webcam). Gesture input was triggered by python programming language as shown by the following code. The first section of the code shows how various needed libraries (tensforflow, tflearn, numpy and cv2) were imported for initial dataflow and loading of webcam interface images.

```
import tensorflow as tf
import tflearn
from tflearn.layers.conv import conv_2d,max_pool_2d
from tflearn.layers.core import input_data,dropout,fully_connected
from tflearn.layers.estimator import regression
import numpy as npnfrom PIL import Image
import cv2
import imutils
def main():
    aWeight = 0.5
    # get the reference to the webcam
    camera = cv2.VideoCapture(0)
    # region of interest (ROI) coordinates
    top, right, bottom, left = 10, 350, 225, 590
    num_frames = 0
    start_recording = False
    # keep looping, until interrupted
    while(True):
        (grabbed, frame) = camera.read()
        # resize the frame
        frame = imutils.resize(frame, width = 700).
```

4.1.2 Gesture Execution

The implemented design thinking framework explicitly takes the sequence of sub-gestures for a given gesture set and uses it to learn a model for subsequent gesture sets providing greater user flexibility for feedback, analytic and output entities. The following code subsection describes gesture execution of hand-palm gesture.

```
def main():

    camera = cv2.VideoCapture(0)

    # hand palm coordinates
    top, right, bottom, left = 10, 350, 225, 590

    # initialize num of hand palm frames
    num_frames = 0
    start_recording = False

    # keep looping, until interrupted
    while(True):
        # get the current hand palm frame
        (grabbed, frame) = camera.read()

        # resize the frame
        frame = imutils.resize(frame, width = 700)

        # flip the  hand palm frame so that it is not the mirror view
        frame = cv2.flip(frame, 1)
        # clone the hand palm frame
        clone = frame.copy()

        # get the height and width of the hand palm frame
        (height, width) = frame.shape[:2]


        # get the ROI of hand palm frame
        roi = frame[top:bottom, right:left]


        # to get the background, keep looking till a threshold is
reached
        # so that our running average model gets calibrated
        if handpalm_frames < 30:
            run_avg(gray, aWeight)
        else:
            # segment the hand region

        hand palm = segment(gestur1)
```

4.1.3 Gesture Improvement Phase

The feedback and analytics entities define incomplete gesture input sets, and find the optimum parameters for ANN topology models by classifying gesture input data from feedback, analytics and output entities into two steps. First step is learning step (training step) in which a classifier is built to describe a predetermined set of new input data classes from the three entities. The second step builds a model done in the first training stage for classification with a classifier accuracy of known data to new alternative gesture input data.

The following code shows how the training and re-train of different predicated new hand palm gesture for user's in the event that of failure of initial gesture implementation. The first part of the code defines the gesture image read immediately creating a model of it and checking if it is well executed. A second definition introduce a new model of hand gesture name swing or fist if the initial hand palm gesture is not well executed.

```
def getPredictedClass():
    image = cv2.imread('Temp.png')
    handpalm_image = cv2.cvtgesture(image, cv2.GESTURE)
    prediction = model.predict([gesture_image.reshape(89, 100, 1)])
    return np.argmax(prediction), (np.amax(prediction) /
(prediction[0][0] + prediction[0][1] + prediction[0][2]))
def showStatistics(predictedClass, confidence):
    textImage = np.zeros((300,512,3), np.uint8)
    className = ""

model.load("TrainedModel/GestureRecogModel.tfl")

    if predictedClass == 0:
        className = "Swing"
    elif predictedClass == 1:
        className = "gesture"
    elif predictedClass == 2:
        className = "Fist"
        # Predict
        image = cv2.imread('Temp.png')
        handpalm_image = cv2.cvtColor(image, cv2.COLOR_BGR2handpalm)
        prediction = model.predict([handpalm_image.reshape(89, 100, 1)])
        return np.argmax(prediction), (np.amax(prediction) /
(prediction[0][0] + prediction[0][1] + prediction[0][2]))
```

4.2 Confusion Data Table Results

The results show that from the 10 test users who recorded hand palm gesture interactions performed in the framework tests recognized 9 times correctly which represents a success rate of 90% when the user is standing performing gestural hand palm images while interacting with the system, 85% success rate were sited and 85% while jumping leading to an overall accuracy of 77% of gestural interaction.

Test users performed hand palm gesture interaction within the design thinking framework when sited, standing or while jumping. The different user body state was in purpose to conclude the best body state for the execution of gestural interaction.

Confusion matrices accounted for error-rate, accuracy, specificity, sensitivity, and precision of the design thinking framework for gesture interaction. The standing user body state presented best execution of hand palm gesture interaction. The experimental results for the design thinking framework of gesture interaction were presented as confusion matrix as shown in the following table 4.1

Table 4.1: Confusion Matrix Data Table Results

User Body State	**Sitting**	**Standing**	**Jumping**	**Accuracy**
Sitting	85%	0	0	0
Standing	0	90%	0	0
Jumping	0	0	82%	0
Accuracy	0	0	0	77%

4.3: Evaluation Results

The average user time used, number of gestures interaction sets, average execution time and frequency of gesture execution were variables used to determine the significance or the framework. The following table 4.2 presents the evaluation results of comparison with user defined gestures for surface computing abbreviated as UDGSC

Table 4.2: Evaluation Results

	Average user time used (Seconds)	**Number of gestures interaction sets**	**Average execution time**	**Average gesture execution. (seconds)**
Eva 1				
UDGSC	16.333	2	3.166	.9.05
Current Framework	10.55	5	2.100	
Eva2		2	8.111	1.357
UDGSC	16.333	5	2.100	
Current Framework	7.00			
Eva3				
UDGSC	14.333	2	7.111	
Current Framework	5.555			1.444
		5	2.100	
Eva4				
UDGSC	16.333	2	8.111	
Current Framework	8.444	5	2.100	1.357

The average user time turn around reduced compared with framework for user defined gestures for surface computing by a half of the initial time showing ease to use by the users. The average gesture input sets increased up to 75% with average execution time reducing by 82% with the interaction. The user's average gesture input sets increased up to 75% with average execution time reducing by 82% with the interaction. Frequency of execution increased with the framework simulation.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

5.1 Summary

The research modelled, implemented and tested a design thinking framework for gesture interaction framework based on three core entities: gesture output, analytics and feedback. The entities improve the information provided by the device about user input. Each entity represents gesture input and output activities that can be recognized from samples and performs updates on gesture state.

The feedback and analytics entities define incomplete gesture input sets, and find the optimum parameters for ANN topology models by classifying gesture input data from feedback, analytics and output entities into two steps: The learning step (training step) and model retraining where in which a classifier is built to describe a predetermined set of new input data classes from the three entities. The second step builds a model done in the first training stage for classification with a classifier accuracy of known data to new alternative gesture input data

The design thinking framework is generic and cuts across different gesture interaction computing from personal computing (gestural swiping of documents and files to quickly process desired output without touching Pc or phone), entertainment (for gaming to manipulate body activity scenarios as input).

The results show that from the 10 test users who recorded hand palm gesture interactions performed in the framework tests recognized 9 times correctly which represents a success rate of 90% when the user is standing performing gestural hand palm images, leading to an overall accuracy of 77% of gestural interaction while interacting with the system.

5.2 Recommendations for Further Study

Even though the findings show that the design thinking framework provides gesture interaction design guidelines which include addition of three core entities: analytics, feedback and output; the research did not go further to combine other low-level user's characteristics for instance; end users who are limited to one hand only or cannot use both hands.

It would be interesting to extend the low-level user characteristics guidelines to cover specific levels of gesture interaction usability. Finally, the research recommends the adaption of the classification of three categories of the design thinking framework provides gesture interaction design guidelines which include addition of three core entities: Analytics, Feedback and Output.

Future studies can improve the design thinking framework by considering adaptive gesture interaction systems since they are more creative, simple and flexible for end users as compared to the normal gestures that lock users to specified gestural body movements.

REFEERENCES

Alsheakhali, M., Skaik, A., Aldahdouh, M., & Alhelou, M. (2011). Hand Gesture Recognition System. *Computer Engineering*, 5. https://doi.org/10.1021/ac00094a004

Angelini, L. (2017). *A framework for abstracting, designing and building tangible gesture interactive systems*.

Baert, P. (2015). *the Role of Design Thinking Making People Want Things People*.

Billinghurst, M. (2015). *Gesture Based AR Interaction Research at the HIT Lab NZ*.

Charachon, A. (2013). Traitement endoscopique du diverticule de Zenker. *Acta Endoscopica*, *43*(3), 105–108. https://doi.org/10.1017/CBO9780511536328.010

D.School. (2013). *An introduction to Design Thinking*. 1–15. https://doi.org/10.1007/978-1-4302-6182-7_1

Dalka, P., & Czyzewski, A. (2010). *Human-computer interface based on visual lip movement and gesture recognition*. *7*(3), 124–139.

Fritschi, M., Buss, M., Drewing, K., Zopf, R., & Ernst, M. O. (2004). Tactile Feedback Systems. *Workshop at the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004).*, 1–8. http://www.is.tuebingen.mpg.de/fileadmin/user_upload/files/publications/TWF1.pdf

Kekong, P. E., Ajah, I. A., & Chidiebere, U. (2021). *Real Time Drowsy Driver Monitoring and Detection System Using Deep Learning Based Behavioural Approach*. *9*(1), 11–21.

Kimbell, L. (2012). *Rethinking Design Thinking : Part 1*. *3*(3), 1–27.

Lee-Cosio, B. M., Delgado-Mata, C., & Ibanez, J. (2012). ANN for Gesture Recognition using Accelerometer Data. *Procedia Technology*, *3*, 109–120. https://doi.org/10.1016/j.protcy.2012.03.012

Morra, L., Delsanto, S., & Correale, L. (2019). Introduction to Deep Learning. *Artificial Intelligence in Medical Imaging*, 15–45. https://doi.org/10.1201/9780367229184-2

Morris, T. (2000). *Gesture Recognition*. 121–136. https://doi.org/10.1007/978-1-4471-0455-1_9

Muser, S. (2015). *Gestures in Human-Computer-Interaction*.

Nehaniv, C. L. (2005). Classifying types of gesture and inferring intent. *Companions: Hard Problems and Open Challenges in Robot-Human Interaction*, 74. https://doi.org/ng

Owen, C. L. (2006). Design Thinking : Driving Innovation. *BPM Strategies Magazine*, 1–5. BPMInstitute.org

Palacios, J. M., Sagués, C., Montijano, E., & Llorente, S. (2013). Human-computer interaction based on hand gestures using RGB-D sensors. *Sensors (Switzerland)*, *13*(9), 11842–11860. https://doi.org/10.3390/s130911842

Plattner. (2009). *Design thinking*. 240. https://doi.org/10.1145/2535915

Sharma, N., Raj, A., Kesireddy, V., & Akunuri, P. (2021). *Machine Learning Implementation in Electronic Commerce for Churn Prediction of End User*. *5*, 20–25. https://doi.org/10.35940/ijsce.F3502.0510521

Sharma, R. P., & Verma, G. K. (2015). Human Computer Interaction using Hand Gesture. *Procedia Computer Science*, *54*, 721–727. https://doi.org/10.1016/j.procs.2015.06.085

Singla, S., & Patel, A. (2020). *Comparative Study of the Deep Learning Neural Networks on the basis of the Human Activity Recognition*. *8*(11), 27–32.

Steinke, G. H., Al-deen, M. S., & Labrie, R. C. (2017). Innovating Information System Development Methodologies with Design Thinking. *5th International Conference on Applied Innovations in IT*, *March*, 51–55. https://doi.org/10.1108/13563280110409872

Trigueiros, P. (2013). *Hand Gesture Recognition System based in Computer Vision and Machine Learning : Applications on Human-Machine Interaction*.

Tse, E., Shen, C., Greenberg, S., & Forlines, C. (2006). Enabling interaction with single user applications through speech and gestures on a multi-user tabletop. *Proceedings of the Working Conference on Advanced Visual Interfaces - AVI '06*, 336. https://doi.org/10.1145/1133265.1133336

Visell, Y., & Cooperstock, J. (2007). Modeling and Continuous Sonification of Affordances for Gesture-Based Interfaces. *Proceedings of ICAD 2007, January 2007*, 423–429.

Wobbrock, J. O., Morris, M. R., & Wilson, A. D. (2009). *User-Defined Gestures for Surface Computing*.https://www.microsoft.com/en-us/research/wp-content/uploads/2009/04/SurfaceGestures_CHI2009.pdf

Young, G., & Consultant, P. (2010). *Design thinking and sustainability*. *61*(0), 1–27.

Zhenhui, A. J., Chan, A., Chuan, H., Phan, A., Quang, T., & Hassanein, K. (2017). *THE EFFECTS OF GESTURE-BASED AN EMBODIED VIEW*.

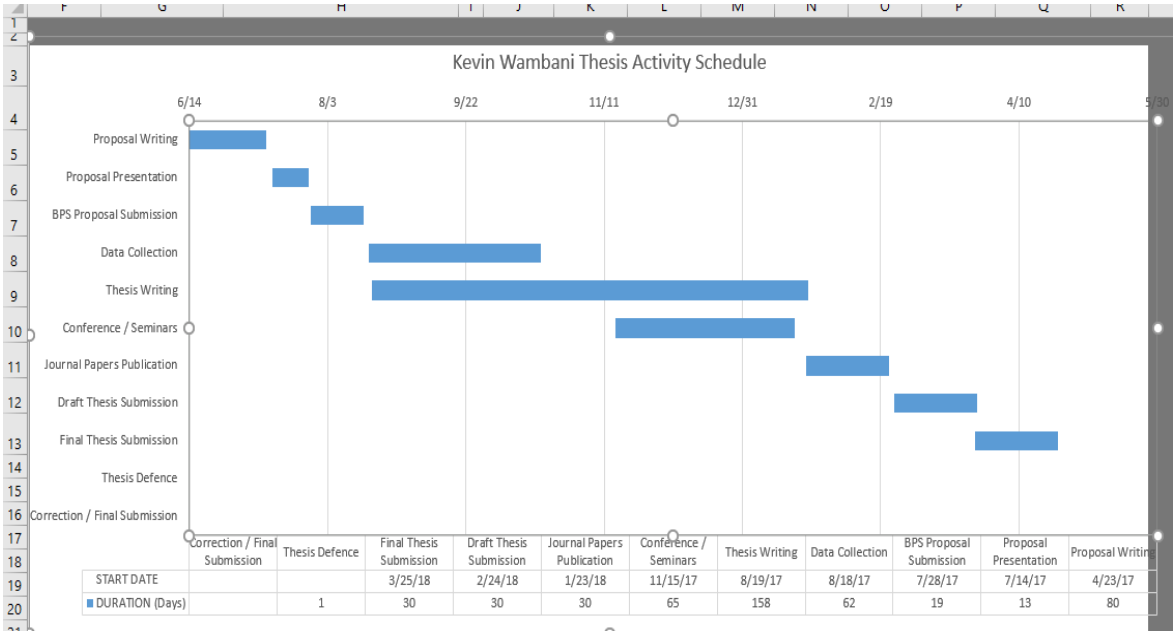
APPENDICES

Appendix I: Cost and Materials

S. No	Items	Specifications	Quantity	@Kshs	Amount Kshs
1.	Materials and Supplies	Printing Papers Printing Cost Questionnaire Forms Report materials Communications	105	700	90,000 45,000
3.	Logistics	Field Work Professional Conferences Sponsor Meetings Consultants travel			800,000
4.	Equipment's Hire	Testing Devices renting	15 devices		535,000

5.	Services	Data Storage Framework testing Publication Cost Photography Duplication of reports Data Purchase			650,000
	TOTAL				**2.120,000**

Appendix II: Activity Schedule (Gantt Chart)



	Correction / Final Submission	Thesis Defence	Final Thesis Submission	Draft Thesis Submission	Journal Papers Publication	Conference / Seminars	Thesis Writing	Data Collection	BPS Proposal Submission	Proposal Presentation	Proposal Writing
START DATE			3/25/18	2/24/18	1/23/18	11/15/17	8/19/17	8/18/17	7/28/17	7/14/17	4/23/17
DURATION (Days)		1	30	30	30	65	158	62	19	13	80

Appendix III: Gesture Recognizer Code

```
    import tflearn
    from tflearn.layers.conv import conv_2d,max_pool_2d
    from tflearn.layers.core import
input_data,dropout,fully_connected
    from tflearn.layers.estimator import regression
    import numpy as np
    from PIL import Image
    import cv2
    import imutils


    # global variables

    def resizeImage(imageName):
        basewidth = 100
        img = handPalmImage.open(imageName)
        wpercent = (basewidth/float(img.size[0]))
        hsize = int((float(img.size[1])*float(wpercent)))
        img = img.resize((basewidth,hsize), Image.ANTIALIAS)
        img.save(imageName)


    def run_avg(image, aWeight):
        global bg
        # initialize the background
        if bg is None:
            bg = handPalmImage.copy().astype("float")
            return


        # compute weighted average, accumulate it and update the
background
        cv2.accumulateWeighted(image, bg, aWeight)


    def main():
        # initialize weight for running average
        aWeight = 0.5


        # get the reference to the webcam
        camera = cv2.VideoCapture(0)

        # initialize num of frames
        num_frames = 0
        start_recording = False

        # keep looping, until interrupted
        while(True):
            # get the current frame
```

```
            (grabbed, frame) = camera.read()

            # resize the frame
            frame = imutils.resize(frame, width = 700)


            # flip the frame so that it is not the mirror view
            frame = cv2.flip(frame, 1)

            # clone the frame
            clone = frame.copy()


            if num_frames < 30:
                run_avg(handpalm, aWeight)
            else:
                # segment the hand Palm region
                hand = segment(handpalm)


                # check whether handPalmImage region is segmented
                if hand is not None:
                    # if yes, unpack the thresholded image and
                    # segmented region
                    (thresholded, segmented) = handPalmImage


            # draw the segmented hand
            cv2.rectangle(clone, (left, top), (right, bottom),
(0,255,0), 2)


            # increment the number of frames
            num_frames += 1


            # display the frame with segmented hand
            cv2.imshow("Video Feed", clone)


            # observe the keypress by the user
            keypress = cv2.waitKey(1) & 0xFF


            # if the user pressed "q", then stop looping
            if keypress == ord("q"):
                break

            if keypress == ord("s"):
                start_recording = True


    def getPredictedClass():
```

```
        # Predict
        image = cv2.imread('Temp.png')
        handpalm_image = cv2.cvtColor(image,
cv2.COLOR_BGR2handpalm)
        prediction = model.predict([handpalm_image.reshape(89,
100, 1)])
        return np.argmax(prediction), (np.amax(prediction) /
(prediction[0][0] + prediction[0][1] + prediction[0][2]))


    def showStatistics(predictedClass, confidence):


        textImage = np.zeros((300,512,3), np.uint8)
        className = ""

        if predictedClass == 0:
            className = "Swing"
        elif predictedClass == 1:
            className = "Palm"
        elif predictedClass == 2:
            className = "Fist"


        cv2.putText(textImage,"Pedicted Class : " + className,
        (30, 30),
        cv2.FONT_HERSHEY_SIMPLEX,
        1,
        (255, 255, 255),
        2)

        cv2.putText(textImage,"Confidence : " + str(confidence *
100) + '%',
        (30, 100),
        cv2.FONT_HERSHEY_SIMPLEX,
        1,
        (255, 255, 255),
        2)
        cv2.imshow("Statistics", textImage)

    main()
```