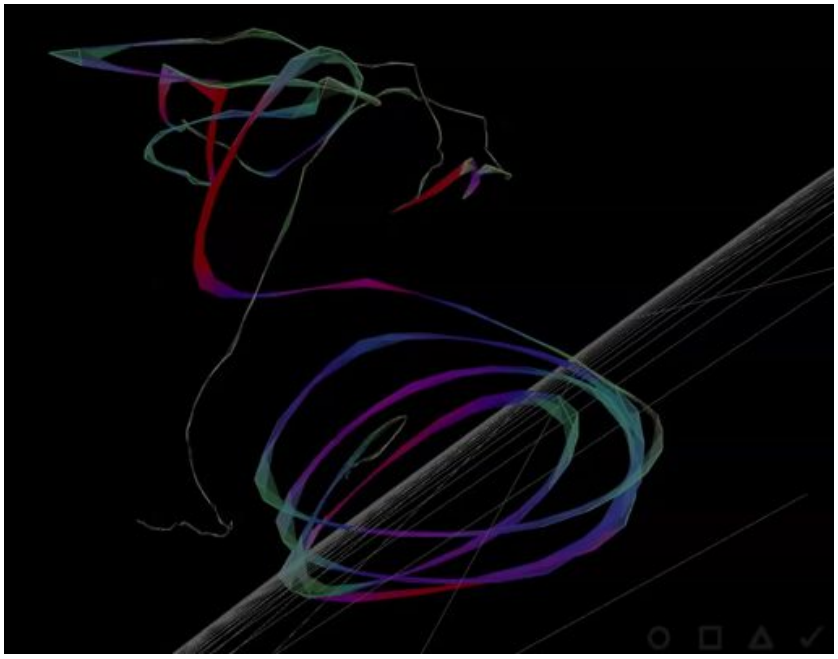
TOWARDS A NATURAL USER INTERFACE FOR VISUALISATION

BEN ALUN-JONES



Analysis and implementation of gestural recognition methods for collaborative real time content control and creation

Department of Electrical and Electronic Engineering

Imperial College London

June 2010

ABSTRACT

The twin developments of more mobile computing power and 3-D display technology mean that the requirements for control of computer systems are changing. The keyboard and mouse, the paradigm of the office, do not afford the intuitivness that is required to operate these new systems effectively. New methods need to be investigated that provide a more natural way of interacting. Gesture recognition is one of the potenial methods which could create a more natural interaction style. In order to test that hypothesis, a system was built that afforded both touchscreen based gestures, and physical gestures in the context of control of a 3-D environment. The steps in creating that system and the results of user testing with the system are presented.

Words represent your intellect.
The sound, gesture and movement represent your feelings.

— Patricia Fripp

ACKNOWLEDGMENTS

The scope and context of this project would not have been possible had I not had the opportunity to spend my final year at UC Santa Barbara. I would like to thank all at Imperial who gave me this once in a lifetime opportunity, in particular Prof. Alessandro Astolfi and Adrian Hawksworth.

At UCSB, my supervisor, Matthew Turk and George Legrady have provided invaluable help throughout the year. Big thank yous are also in order for Matthew Wright, Charlie Roberts and Syed Reza Ali for help on angles, approaches and iPhones as well as pointing me to the free bagels. Finally I would like to thank all the students, staff and friends I have made here who have inspired me and made this a great year!

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

UI User Interface

GUI Graphical User Interface

NUI Natural User Interface

HCI Human Computer Interaction

HMM Hidden Markov Model

IDE Integrated Development Environment

UDP User Datagram Protocol

TCP Transmission Control Protocol

OSC OpenSoundControl

OCGM Objects, Containers, Gestures, Manipulations

OOP Object Oriented Programming

OSI Open System Interconnection

Part I

THE NEED FOR CHANGE

1 INTRODUCTION

Towards a Natural User Interface for Visualisation

Often there comes a time where things need to change. Such a moment is fast approaching in Human Computer Interaction. For many years now the paradigms of the previous generation, the keyboard and the mouse, have been struggling to keep up with the latest developments in computer technology. Supreme in the office and the laboratory, the practicalities of keyboards and mice in mobile or three dimensional applications is questionable. This thesis aims to outline these new challenges facing Human Computer Interaction, such as large scale visualisations and three dimensional environments. Then, attempting to address these problems, I will explain the design of a novel multi-user system which I have created called *Gesture*.

The starting point however for investigating alternatives to keyboard and mouse centred design is understanding what made them so successful in the first place. It is extremely unlikely that anyone will develop a better system for text entry and navigation of 2-D Graphical User Interfaces (GUIs) than the keyboard and mouse, but as the new age of 3-D computing and visualisation fast approaches, this new problem set needs new solutions.

One main way in which HCI systems need to change in is their intuitiveness. Computers are not easy to use, you have to learn many skills of high cognitive load in order to be able to use them effectively. By making interaction more natural and tailoring it to the way in which we, as humans, behave can reduce the mental challenge of learning these new systems. This offers the potential to widen access of computers and their more useful, yet complex, functionality to a wider range of people and places.

This new paradigm of intuitive, tangible interfaces is refered to as the Natural User Interface (or NUI) [13]. This is a set of rules and interaction styles defining a new style of interaction with computers. The researcher Roope Raisamo describes natural interaction in the following terms:

> *"People naturally communicate through gestures, expressions, movements, and discover the world by looking around and manipulating physical objects; the key assumption here is that they should be allowed to interact with technology as they are used to interact with the real world in everyday life, as evolution and education taught them to do. In addition to the traditional features of interactive digital media, like updatability, freedom of users' to follow their curiosity and interests, natural user interfaces feature a new architectural aesthetics about how to move computation to the real world, creating immersive experiences that involve people senses in the physical space."*
>
> *Roope Raisamo[37]*

Gestures are a crucial part of the NUI as they make use of people's inate abilities to learn kinesthetically, taking advantage of muscle memory, as well as more classical styles such as listening or reading.

Furthermore, people understand systems better if they can see the effect of the steps and actions they take[33]. This is a fundamental part of the design principle known as Direct manipulation[41]; the concept that computer functionality should be represented as distinct objects which can be physically affected by user input.

My new system, *Gesture*, explores the potential for this more natural form of interaction, by posing it in the context of environments for collaborative content creation and manipulation. The system allows the users to make a three dimensional sketch and then move through this creation at will. In this way, the relationship of a user and the virtual environment can be explored. *Gesture* is controlled using two iPhones (but this could be easily extended to a much larger number within the framework). Using mobile phones provides a physical, tangible device which allows the user to both express themselves via the touchscreen or spatially using the accelerometer. Both styles make use of gestures, which are non-verbal actions performed by people to communicate information through motion or expression[19]. Gesture recognition is a complex problem to solve, but solving it will add to the intuitiveness of future Human Computer Interaction methods and will further enhance people's natural view of computers as real social actors[38].

1.1 PROJECT AIMS

The main issues that this project aimed to investigate were as follows:

- `Gesture Recognition:` The main part of this project was the investigation of gesture recognition techniques for practical applications. Many systems have the potential to use gestures for user interface design but few do. By implementing various types of gesture recognition and placing them in a real, useful context, the reasons behind this can be investigated.

- `3-D Navigation:` Navigation in 3-D space is still an unsolved problem. The mapping of controllers are often poor and if a system does not have good navigation control (and is sufficiently immersive for the user) then it can result in serious nausea.

- `Content creation and manipulation:` Realtime content creation is a interesting challenge especially when that data is capable of being manipulated. *Gesture* aims to map the users motion as a means to create a visual model which can then be investigated and changed by the users.

The implementation, research context and results of each of these aims will be presented and discussed in this report in the context of a multi-user collaborative virtual environment as well as discussing the findings from the two public demonstrations of *Gesture* at a UCSB open day and the Media Arts and Technology End of Year Show.

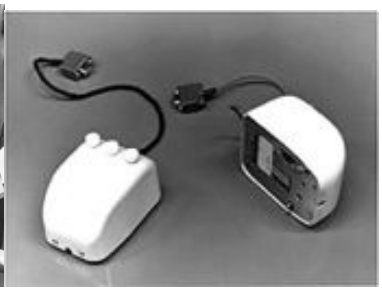
2 THE USER INTERFACE

The challenge for computer systems has always been how to make them interactive in such a way that people can pick them up and make use of computer with little or no instruction. In this chapter we will review the key steps in the history of HCI as well as future developments that point the way to the future directions for man machine interfaces.

2.1 THE DEVELOPMENT OF THE USER INTERFACE

The user interface has been through many changes and transformations since the early days of punch card computing. From Vandevar Bush's first imaginations of the Memex (a thought experiment of a mechanical computer with many of the features of a modern desktop) to Tim Berners-Lee and the rise of the internet, the thoughts of what could be achieved have changed dramactically.

(a) Ivan Sutherland's Sketchpad



(b) The First Production Mouse

(c) The Xerox Star

(d) Apple iMac

Figure 1: Stages in the development of the User Interface

Ivan Sutherland produced what is probably the first user interface as part of his PhD thesis, Sketchpad Sutherland [45]. It had a light pen (the precursor for the mouse) which allowed you to draw on the screen and was the first interface where clicking and dragging was used as an input metaphor. This was to form the inspiration for many of the ideas of direct manipulation many years later.

The next major step forward was created by Doug Engelbart. Engelbart produced the NLS (oN-Line system) which he famously demon-

INTERFACE	ANALOGY	ELEMENTS	ATTRIBUTES
Command Line Interface	Typewriter	Prompt, Command and arguments, Result	Single task, Single user, Command oriented, Keyboard input
Graphical User Interface	Papers arranged on a desk	Windows, Icons, Menus, Pointer (WIMP)	Multi-task, Single user, Task oriented, Keyboard + Mouse input
Natural User Interface*	Objects	Objects, Containers, Gestures, Manipulations (OCGM)	Multi-task, Multi-user, Object oriented, Touch input

Table 1: Comparison of Interface paradigms

strated in 1968. His demonstration showed the first examples of the mouse, computer networking, video calling, collaborative computer work, realtime display editing and the windowing environment. He also was the creator of the first mouse. The first moue to enter production based on this design is shown in Figure 1b.

The next stage was the development of the personal computer. In 1968, Alan Kay conceived the Dynabook, the first conception of a 'personal' computer in the same year Engelbart demonstrated the NLS. This was the motivation for many of Dr. Kay's later developments such as object-oriented programming (OOP). Since the Dynabook was fundamentally a laptop, and far too advanced technology for its time, it is only recently with the release of the iPad that many of his ideas have been realised. Instead, Kay helped produce the Xerox Alto and the Xerox Star (shown in 1c). This was the first machine to make use of the true graphical user interface that we know today. The Star was designed using extensive user testing which helpin to develop many features, some in use today, many still to be widely accepted such as total object-orientation of the operating system.

The final image (Figure 1d) shows the 2010 model Apple iMac. This has nearly all the features imagined over fourty years previously and is testament to the foresight of those individuals. These designs have stood the test of time and are so pervasive that they will likely continue for many years to come.

The question then is how are we to move on with the user interface?

2.1.1 *Understanding the exisiting paradigms*

There have been two major interface paradigms in personal computing. These are the command line interface and the graphical user interface. Each has their advantages and applications and here we will investigate

what their failures are, what makes them useful and the analogies and mental models they use.

Note: The content of this chapter is just some dummy text. It is not a real language.

THE COMMAND LINE INTERFACE: This is a text based form of control. The user is required to enter their syntactically strict command which defines exactly the operation to be performed. This is extremely fast, as long as you know what to type. In order to use the command line successfully the user needs to memorise all the necessary commands. This obviously requires a large amount of recall of information (or indeed lookup in a manual) and cannot be said to be intuitive. If you do not know how to use it, the design of the interface does not give you many clues. The failure is not the speed or the accuracy. In fact this is the most accurate of interfaces, but it takes a large amount of time and investment to learn how to use it and you must be familiar with it in order for it to be usable.

THE GRAPHICAL USER INTERFACE: The Graphical User Interface (or GUI) was developed at Xerox Parc out of attempts to broaden the access to the computer. In order for design to be deemed natural, users must be able to create clear mental model of how the system works, even if this is far from the truthNorman [33]. The concept employed in the GUI is that of papers on a desk. This is window environment (akin to different pieces of paper) which you can select, drag and drop. This is obviously the dominant paradigm forming the basis of Mac OS X and the Microsoft Windows environment. One issue regarding GUIs are that, again, intermediate to expert users understand where and how to navigate, but often, for beginner users this can be diffcult to grasp. However, the main issue with the GUI is that as we enter a more mobile age with smaller, more mobile computers. Is the GUI, the interface designed for the office environment, the correct one to take us into the next age?

First we need to understand and categorise the types of user interface and their features. This will allow us to understand what has been implemented and what is there yet to be achieved. This categorisation is shown in Table 1

2.1.2 *Designing the next step*

In order to look more closely at the design strengths and failures of each inteface, it is important that we look at Shneiderman's eight golden rules of interface design taken from his book, Designing the User Interface[42]. These are:

“

1. **Strive for consistency.** Consistent sequences of actions should be required in similar situations and consistent commands should be employed throughout.
2. **Enable frequent users to use shortcuts.** As the frequency of use increases, so do the user's desires to reduce the number of interactions and to increase the pace of interaction. Abbreviations, shortcuts and hidden commands are very helpful to an expert user.

3. **Offer informative feedback.** For every operator action, there should be some system feedback. For frequent and minor actions, the response can be modest, while for infrequent and major actions, the response should be more substantial.

4. **Design dialog to yield closure.** Sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions gives the operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans and options from their minds, and an indication that the way is clear to prepare for the next group of actions.

5. **Offer simple error handling.** As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple, comprehensible mechanisms for handling the error.

6. **Permit easy reversal of actions.** This feature relieves anxiety, since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data entry, or a complete group of actions.

7. **Support internal locus of control.** Experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Design the system to make users the initiators of actions rather than the responders.

8. **Reduce short-term memory load.** The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions. "

We can see from this breakdown that the command line interface obviously invalidates some of these rules. Feedback is often lacking, some operations are often irreversable and so input errors cannot be adequately handled. However, the main failure as stated above is the fact that in order to use it effectively, you must retain numerous codes, or sequences of actions in your memory in order to use it successfully. There are some GUIs that follow these guidelines successfully, but there are many more that do not.

However, there is a single reason for the unsuitability of the GUI and command line for the next stage of computing, these were paradigms created for the office environment. This move towards a more natural computing style is necessary when we consider three dimensional displays and mobile computing. As the pervasiveness of computing increases and computers or microprocessors appear in unexpected places, then the real need for a new paradigm to address these challenges becomes obvious. 3-D navigation is obviously another huge challenge that has not been adequately addressed by the existing computing paradigms. It is wrong to say the existing paradigms should be removed or retired, they still apply, but mobile and 3-D interfaces cannot be controlled optimally using these systems.

What would happen if the methods we interacted with computers were so obvious, there would almost be no need to recall anything? That is to say the pattern of behaviour needed to interact would be

exactly the same pattern that we use to interact with the real world. This is the concept behind the natural user interface.

2.2 THE NATURAL USER INTERFACE

The Natural User Interface (or NUI) is an interface paradigm which becomes invisible to the user after learning how to interact with it, or ideally, before you ever begin. This depends on computers being able to recognise people's gestures, motions or even thoughts. In order to make this a reality, several new sensor or motion recognition procedures would need to be developed (and at much higher level of accuracy) so that it does become truly seamless. Microsoft is already making large advances in this field with the development of Surface and the imminent release of Project Natal. "Surface" is a touch screen table top interface which provides a new form of 2D interaction and Natal is a optical motion capture system for the XBox 360. This would allow you to move without any physical connection and be able to control a computer game environment.

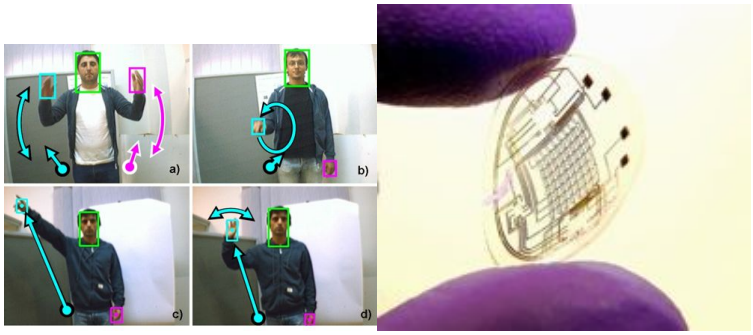
A new theory which encompases natural user interfaces is breaking the key parts of a NUI into Objects, Containers, Gestures and Manipulations (or OCGM) George and Blake [13]. OCGM is pronounced Occam, based on Occam's Razor, the theoretical concept that "entities must not be multiplied beyond necessity"[14] and this breakdown aims to define the interface in terms of the simplest forms of user interaction. George and Blake show that the skills required to operate an OCGM based interface are developed at nine months old, significantly earlier than icon recognition (three and a half years) and the required reading level was not until school age. OCGM is fundamentally a set of abstract metaphors "that support many layers of concrete metaphors". i.e. these are the building blocks for a new interface paradigm and in fact describe the exisitng paradigms as well. OCGM is the description by which we can describe all interfaces as it is the most basic understand of what is required in order to create an interface.

2.2.1 *Human-Computer Interaction and 3-D Visualisation*

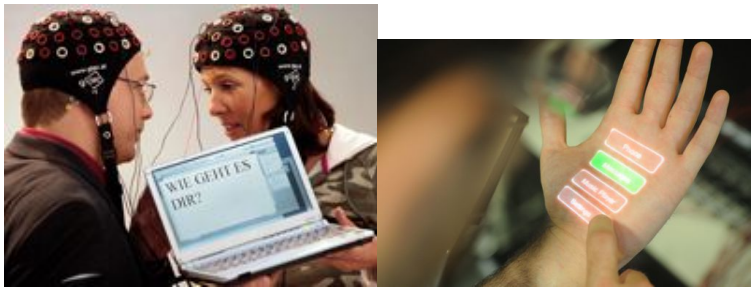
Throughout the development of 3-D Visualisation systems and in particular immersive systems such as CAVE type environments or the Allosphere , the Human Computer Interaction of these environments is often neglected, even though (and perhaps because) it is a difficult and challenging problem. Many things that we do easily in the real world are complex in 3D virtual worlds, Navigation and Object Selection being two classic examples. The challenge of natural interface design for 3-D worlds is made more complex due to the fact that a lot of extremely accurate models based on physics are in fact difficult or complex to use in the real world [34].

Built at UCSB, the Allosphere is one of the world's only corner free 3D environments [17]

Research has been done on 3-D Navigation including taxonomy of techniques used[46]. These point to a still emerging field where even the input device, let alone the characteristics have been determined. 3-D Navigation ultimately will be most successful when the state of the user is fully known, either through an array of cameras or sensor networks. This allows the state of the user to be fully determined and so their intention can be percieved (probably after some fairly advanced machine learning techniques are then applied).



(a) Camera based Gesture recognition[3] (b) Contact Lens based video displays [2]



(c) Brain Computer Interfaces[1] (d) 'Skinput' a novel user interface[16]

Figure 2: Future Directions for the User Interface

2.2.2 *Future input devices*

The array, complexity and quirkyness of input devices show a huge range of options and potentials. From computer vision techniques (which show some of the strongest promise - Figure 2a) to brain interfaces and bio sensing (Figure 2c), the way in which we enter data into a computer system is about to change forever. More early stage research technologies such as Skinput (Harrison et al. [16]) suggest a future where a computer system is built entirely on your body. Finally, augmented reality applications where computer graphics are overlaid on the real world, are another step towards a more ubiquitous computing world. One potential device that would have seemed far fetched or unbelievable in the past is the integrated display on the contact lens (Figure 2b). This also shows that with increased minaturisation, this possible integration of computers and the human body is important, if for no reason other than the usability of such small devices.

2.3 SUMMARY

The history of HCI is one of idea generation and implementation. It was only by exploring the potential of new applications of computing that we came to a better understanding of how we interact with technology. It is this experimentation and new direction that is forcing us again to take a new look at our means of interaction with computers. The plethora of three dimensional applications (made popular by the success of 3-D films such as "Avatar" etc) and the increased portability of technology is taking us towards a new paradigm, one where the user is

king and the computer must understand the user; not one where the user must understand the computer.

Whilst I cannot hope to solve these problems, in the next section I will outline the multi user visualisation environment that I created to attempt to investigate users' behaviour and the reasons for and future steps needs to solve these problems.

Part II

BUILDING GESTURE

3 A 3-D INTERACTIVE MULTI-USER ENVIRONMENT

Many challenges exist for 3-D interactive environments. Actions we take for granted in real life, such as object selection or simply moving through 3-D space, are complex to achieve in virtual worlds. Making a natural interface where you can multi-task (such as selecting an object and navigating at the same time) is a harder problem by a level of magnitude if not more. As we saw in the last section, there is a need for new interaction styles for 3-D environments that will be more natural and better suited. Some of the challenges to overcome are the means of control, the method of feedback, the level of immersion and the accuracy of the system, as well as ensuring that the interface obeys the "golden rules of interface design" (see Section 2), which still applies for 3-D environments in the opinion of the author. Furthermore, by making the system multi-user, we can explore the potential of collaborative environments which allows multiple tasks to be achieved at once. In this section I will outline *Gesture*, the system I have created to investigate some of these interaction styles and the response of users who are controlling it and will highlight some of the key parts that make this system work.

3.1 GESTURE

When constructing *Gesture*, my aim was to create an environment to investigate the potential and challenges of a natural user interface for three dimensional visualisation. This means a system which can support gestural recognition (i.e. one that could recognise natural motion as input), 3D navigation, multiple users and realtime content creation and manipulation. The functional blocks to create such a system and their inter-relations are described below:

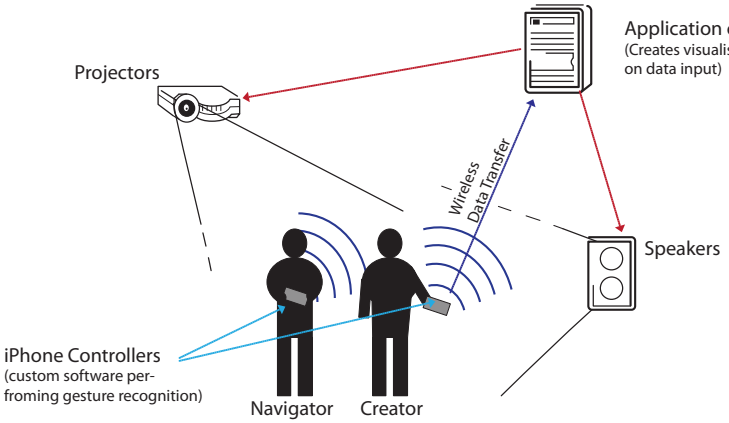


Figure 3: Gesture system overview

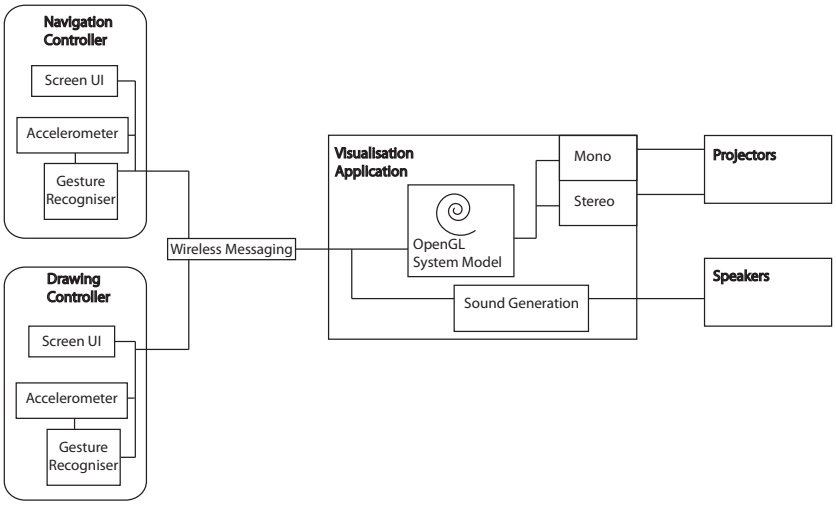


Figure 4: Functional Diagram of Gesture

3.1.1 *System Components*

The system overview shows some of the key features. Two users, one assigned the navigation role and another who is the content creator and manipulator, are looking at a projected environment (either a standard projection or in stereo). There is also sound reproduction which responds to user input adding another modality to improve the user experience.

One obvious means of control for such a system is a mobile 'smartphone' such as an iPhone. These are wireless, reasonably powerful, portable computers which have a range of sensors suited for gesture recognition and natural unencumbered motion. The accelerometer can be used for 3-D gesture recognition and touchscreens provide a means to record 2-D gestures (N.B. Gesture recogntion is discussed in depth in the next section).

Another challenge is then building an intuitive interface around this controller. The controls themselves must be easy to interpret and use and the design of the touch screen UI must be clear and understandable for the user. A means of valid feedback must also be provided (such as the vibration of the phone or audio in the form of earcons).

Earcons are a clear, defined sound which represents an event in a user interface e.g. text message notification

The complete interaction sequence for the system also needs to be mapped out and planned i.e. what form of interaction will take you where and how these gestures will change your environment, so that errors can be prevented and the user maintains control of the system at all stages.

The data flow also needs to be controlled; in particular the wireless networking from the phone controllers to the visualisation system. Phone data needs to sorted into 'packets' so it can be transfered and the visualisation system needs to read and interpret this data for projection and sonification.

The Allosphere at UCSB is one of only three corner-free multi-user virtual environments to have been built making it one of the most immersive 3-D environments in the world.

In order to create a 3-D environment, OpenGL was used to provide a way to create custom graphics which can easily be extended to true stereoscopic 3-D for an environment such as the AlloSphere as well as standard display equipment. A camera for this 3-D space also needs to be created which can be controlled by the user. The visualisation application also needs to handle the sound notifications and generation.

These sounds take the form of either earcons which respond to the gestures or sound which respond to the data received as part of the visualisation. Finally the application also needs to output the data to the projectors and speakers to complete this multi-modal visualisation environment.

3.1.2 *Next Steps*

In the next section, the construction of the mobile controller software and how the gesture recognition works will be discussed. Then the visualisation process, including the inspiration and the technical details of creating the OpenGL space and sonic environment, will be explained. Finally I will describe the connectivity of the system and the wireless messaging format used to create and complete the system.

3.1.3 *Related Work*

Many people have posed the question of gesture recogntion as a means for controlling a 3-D environment. One of the earliest such systems was "Put-that-there" which involved the user controlling simple shapes through speech and basic gesture recognition (Bolt [10]) and there has been a lot of work in this field to apply it to visualisation of earth environments (Krum et al. [22]) or wider 3-D navigation (Krahnstoever et al. [20]). Some form of gesturing is also used for sketching. It has been desireable since the advent of computers (Landay and Myers [23]) and has been applied to 3-D sketches more recently (Zeleznik et al. [53]). Research is now starting into mobile technology as a means for control of virtual environments (Kela et al. [18]), but none found combine navigation, mobile control and content creation in the same way as *Gesture*.

4 BUILDING THE CONTROLLER

Ensuring the user has a natural means to interact with the system was a crucial step in the project. In order to do that a wireless controller which could be used for some sort of gesture recognition was desirable. Users already have an affinity with mobile telephony. For many, it is the most intimate piece of technology they own, which provides personal access to their friends and family. Hence it made sense to use a controller which most users already had some familiarity with.

4.1 AN IPHONE-BASED CONTROLLER

The iPhone platform was chosen for the hardware of the controller. With a very responsive touch screen and built-in accelerometer, 2D and 3D gestures can be mapped. It also has, unlike many phone development platforms, good user support and an excellent integrated development environment which allowed rapid prototyping making it a good, all round choice for the controller.

Two custom Objective-C applications were written to address each of the user roles for the visualisation. One to navigate through the 3-D environment and one to generate the content for this exploration. The operation of each was abstracted as far as possible from a knowlege of the visualisation so that they were simply controllers and could be applied to other visualisations or even other systems. Hence, the controllers needed to solve the following requirements:

4.1.1 *Requirements of the controller*

- **The controller needs to be intuitive to use and as natural as possible.** This requires a clear interface design on the touchscreen and a strong relationship between the actions performed and the impact that has on the visualisation. Naturalness is an issue for this kind of environment as expert users may have different perceptions to beginners (which indeed was the case with the user testing performed).
- **The recognition rates should be as fast as possible to ensure actions are relevant.** If the timelag between a gesture being performed and the response is too long, it may seem like there is no correlation between the interaction and the action performed. Large lag times also impact the usefulness of the technology and the potential for multiuser collaboration as the response of various actions may be confused.
- **Gesture recognition should be done on the phone.** In order to ensure that separation of the visualisation environment and the controller is complete, the controller should just send the name of the gesture performed, not raw data which needs processing. Enforcing this separation also improves the development of a multiuser system and makes it scalable to a large number of users.

(a) The Draw Application Orientation (b) The Draw App Layout

Figure 5: Drawn - Layout and Orientation

4.1.2 *Designing the User Interface*

The two applications, Draw and Navigate, both need to have clear controls as well as leaving room for gestures to be performed. Another part is that due to limitations of the 3-D gesture recognition method used, a button is required to signify the presence of a physical gesture. Both had the same core functionality of the gesture recognition software, but both required slightly different layouts to specialise each for their respective tasks.

Both applications also have a setup screen which allows entry of the required IP address of the visualisation application (This is described in more depth in section6). All the elements were constructed from the built in iPhone UI widget set for consistency.

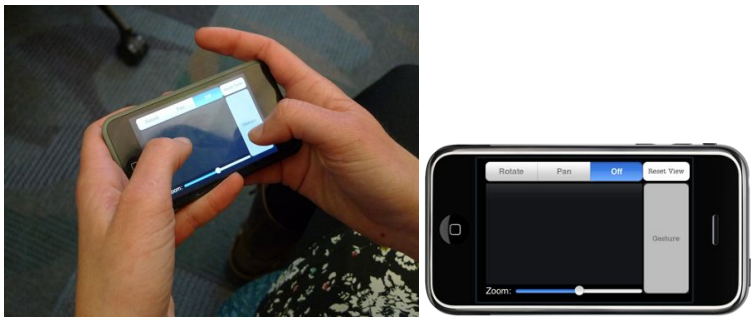
Draw

The Draw app's main functionality is to create the form for the visualisation. This means the application must provide data to the visualisation engine to create a 3-D model yet the aesthetic control of the visualisation must rest with this user. This provides us with a context for designing the layout.

Using the accelerometer data, we can create form based on the user's motions (see next section). Gestures can then be used to change the drawing options and general aesthetic. As stated previously, all the recognition is done on the phone and so this is simply sending a command to change an element (without any prior knowledge of the scene).

The main functional elements are deciding when to draw, when to perform a 3-D gesture (which due to technical limitations requires a button to be pressed) and a reset or clear function. These are clearly displayed with labels describing their function and a large space for drawing gestures5b. The drawing is a switch as it gives a clear indication of the current output state of the controller. The other buttons simply need to behave as buttons, so were kept simple to minimise confusion for the user. The reset button also turns off the draw functionality and provides an easy way for the user to undo a particular action.

The vibration function of iPhones was then used to provide immediate feedback as well as sending a notification to the visualisation environment for audio feeback.



(a) Orientation of the Navigate Application (b) The Navigate App layout

Figure 6: Navigation - Layout and Orientation

The orientation of the screen was chosen to be vertical and in held in one hand since this fitted closest to the analogy of a paintbrush and allowed the user to retain full motion when moving the phone (for drawing).

Navigate

Due to the complexity of 3-D navigation, the Navigate application needed to be more complex in layout than Draw. The main functions for 3-D navigation are:

- **Rotation about the point observed by the camera**
- **Panning** Moving the camera in an x and y direction relative to the camera direction.
- **Zooming in and out of the scene**

Rotation and Panning were chosen to be separate controls since it is rare that you would want to do both at the same time. Zooming is an operation a user would want to do at the same time as rotating and panning, hence it is a separate control. As the control is mostly done through tilting the phone, a two handed orientation was chosen to be more accurate and hence the placement of the large gesture button (for 3-D gestures) and the zoom slider. The large black space is then left to draw gestures on the touch screen.

The operation of the camera is selected from the row of buttons along the top (defaulting to off), but this could be potentially triggered by gestures as well. As it stands, each gestures take the camera to a predefined view to aid with navigation. Like the Draw app, there is a reset button which returns the view to a default location (for example if the user pans to far beyond the scene).

The rotation and panning data, unlike for the drawing data for the Draw application, needs to be processed so it is useful and reduces the nausea inducing capabilities in an immersive 3-D environment. For example, limiting the range of data being sent (having a dead zone where no rotation is performed and a maximum rate) as well as smoothing the inherently noisy accelerometer data were necessary steps for a smooth camera motion.

4.2 GESTURE RECOGNITION

In order to create a more natural user interface, as well as using a mobile platform which instantly liberates the user, gesture recognition forms a key part of the move towards a pattern of interaction more in tune with the human user. Gestures and gesture recognition have been mentioned a lot prior to this section, so here we will explain what a gesture is and how we can go about attempting to recognise gestures in a reliable and repeatable fashion. I will then cover the implementation details of the 2-D and 3-D methods on the iPhone.

4.2.1 *What is a gesture?*

Gestures are non-verbal signals that relate information through motion or expression[19]. For example the way you stand in relation to someone, how you look at them, the way you move your hands when you talk or simply the motion of your eyebrows. But gestures do not include how close you stand to someone, or pure undirected expression of emotion. Gesturing is simply a non-verbal means to convey a specific piece of information to another person. In addition, the fundamental and most interesting part of gesturing is that people only "gesture" when in the presence of another person [19, 6].

The difficulty and complexity of gestures is their ever changing behaviour. Gestures vary based on social background and context, age, gender, culture, location, even among friendship groups. In fact every variable that impacts human speech impacts upon gestures since after all it is non-verbal communication. This means it is an extremely complex problem with few clear cut solutions. Even in psychological literature [Kendon [19], Armstrong et al. [6], McNeill [29]], the complexity of gestures, their real meanings and their significance is hard to define. This poses the question : Should we attempt to create methods to recognize the plethora of human gestures, or should we structure our gesture set and limit it in a similar way to sign language?

The key points about constructing a gestural "language" is that, as said before, different gestures mean different things to different people and people ability to reproduce even known gestures is extremely variable and hard to define. Also the language we create should have some degree of consistency and if possible allow some sort of metaphor to be associated with it. Whilst this is hard with current technology (in particular online implementations of recognition), it is none the less desirable.

4.2.2 *Automatic Gesture recognition*

One of the difficulty of recognising gestures is getting the required data in a machine readable, high quality format, let alone the algorithms to recognise gestures. Focusing on the most basic sets of gestures, they are fundamentally motion and so require you to record some sort of value, be it acceleration, velocity or displacement. This will fundamentally be noisy and so recognition algorithms will need to be robust or feature some form of pre-processing. They are also dynamic and so require some sort of segmentation of time to define the start and end of the gesture.

In two dimensions, many of these problems are solved since the location of the finger on the screen (in x and y coordinates) in accurately recorded. The start and end of the gesture is well defined by the time at which you start and finish touching the screen and the associated noise is minimal.

For three dimensional gestures this is more complex. People do not repeat gestures in that similar a fashion (there is noise in the action, let alone the recording of the action) and time segmentation of the gesture is hard. Luckily, using the iPhone, we can use a button in the user interface which is held down to define the start and end of the gesture. This gives us our data set which we can then investigate methods to recognise gestures automatically using either machine learning methods such as Hidden Markov Models, which are complex or using simpler template matching schemes. Both methods are their pros and cons are outlined in the next section.

4.2.3 *Related Work*

Touchscreen based methods

There have been two main waves of interest into online handwriting recognition for two dimensional drawing applications. There was a large amount of activity in the 50s till the 70s when computers first became popular. The second wave was in the early 90s with the advent of touchscreens which could accurately be used for graphical drawing and is continuing now with mobile devices and the need for text entry on smaller displays.

The development of online and offline methods occured reasonably in parallel for the first stage until, somewhat suprisingly, online methods were shown to be superior for recognition of even the same data (Mandler et al. [25]). Even with the greater complexity of drawing even something simple like an 'E', there are multiple ways of drawing it which would complicate online recognition, but this can be successfully handled and in fact the temporal information can then be used to improve the recognition rates sufficiently. Methods using Shape recognition, Character Recognition, Time sequence based methods, curve matching and other methods show various levels of success for handwriting recognition (and are all outlined thoroughly in Tappert et al. [47] and Plamondon and Srihari [36] who both review the field), but the requirements for *Gesture* imply a simpler and less complex method.

These methods for simple gesture recognition are overblown and overly complex. The simplicity of the shapes required for gesturing means methods can be much simpler and in fact, when implemented on a phone, need to be simple, have minimal memory requirements and return results fast. That is why Wobbrock et al. [51] and their paper "Gestures without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes" is actually extremely effective for simple geometric shape recognition and will be implemented for use in *Gesture.* Full details of implementation are explained below and results are presented in Section 7.

Accelerometer based methods

Touchscreen based methods are relatively mature in comparison for 3-D gesture recognition. Lots of research has been put into computer

vision based methods (detailed more thoroughly in Moeslund and Granum [30], Gavrila [12]). For *Gesture*, the focus was implementing 3-D gesture recognition on a mobile device and so accelerometer based methods are the most appropriate. Acceleromter based methods use either basic accelerometers (Liu et al. [24]), Wii Controller (Schlömer et al. [39], Sreedharan et al. [44]) or more recently extending to mobile platforms too (Vajk et al. [49], Niezen and Hancke [32]). In terms of the recognition algorithms used, Hidden Markov Models were used in (Schlömer et al. [39] again and Mäntyjärvi et al. [26], Ward et al. [50], Kela et al. [18]) which show reasonable success. Some use Neural Network based approaches (Murakami and Taguchi [31], Boehm et al. [9]) and there is also the "$3 Recognizer" developed by Kratz and Rohs [21] as an example of a template matching scheme. The relative advantages and disadvantages show that with continuous HMMs, time segmentation (and so continuous gesture recognition - i.e. no button presses) can be achieved (Elmezain et al. [11]). But as of yet, there have been no implementations of this have been on mobile platforms.

4.2.4 *2-D Gesture recognition - The $1 Recognizer*

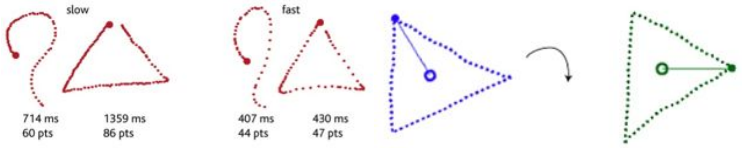
As stated above, there has been a large amount of research into the field of two dimensional gesture recognition. As one of the main requirements for this project was to develop a system where all the gesture recognition was done on the phone, a minimal solution was required. Hence the "$1 gesture recognizer", a touchscreen recognition method developed by Wobbrock et al. [51] was used. This is a template matching algorithm that has comparable recognition rates to more advanced systems, but is significantly simpler and less computationally and memory intensive i.e. perfect for recognition on a mobile device. A disadvantage is that the recognition is done based on a strictly defined gesture set and so no dynamic learning can be performed to tune the system to the user without recording a new set of templates.

The main stages of the algorithm are primarly normalising the data so it can be compared. Those key stages of operation are:

1. Resampling the data to ensure there are the same number of points in the data under test as in the templates.
2. Rotation of the gestures so the samples and templates are of the same orientation
3. Scale and Translate the sample so that the dimensions and centre points are the same as the templates
4. Find the best template match using a Golden Section Search for the rotation of the algorithm and produce a value

Then a theshold can be put on the gestures to ensure that false positives are prevented against to some degree. Each of these stages is now discussed in finer detail.

RESAMPLING: Natural variance in the speed at which a gesture is performed can lead to a variable number of points to compare. This is also influenced by the speed of hardware recording etc, so we want to remove this variability. Figure 7a shows some of variablility in touchscreen points. These points are then resampled



(a) Variance in the number of sample points for touchscreens

(b) Rotation to "indicative" angle

Figure 7: $1 Recognizer - Key steps [51]

to be of size N. All the templates are also of this size to simplify (and speed up) the comparison). The setting of N is obviously a crucial value. N needs to be sufficiently large to distinguish between gestures, but not to large as then upsampling points would introduce unwanted noise in the data. This was found to be 100 points for the iPhone (much higher than the number recorded in the original paper).

ROTATION BASED ON THE INDICATIVE ANGLE: Finding a way to normalise the data based on rotational angle is a complex problem and there is no closed solution. In the paper, after considering numerous methods, they describe a "rotation trick" which is fast and efficient. By taking the centre point or centroid of a gesture and the first point where the gesture began, we can make a good estimation as to the angle normalising the gesture7b. Any small error in this quick estimation is then handled by the Golden Section Search (GSS) in the final stage of the algorithm.

SCALING AND TRANSLATION: The gesture sample is scaled non-uniformly to a square. This means that changes in the sample values when in the final stage the gesture samples are rotated later is due to the rotation only and not the aspect ratio of the points. Once this is done, the samples are then translated back to the origin with the centroid as the centre point.

TEMPLATE MATCHING BY FINDING THE OPTIMAL ANGLE: Firstly the average distance between each point and each of the templates is found using the equation :

$$d_i = \frac{\sum_{k=1}^{N} \sqrt{(C[k]_x - T_i[k]_x)^2 + (C[k]_y - T_i[k]_y)^2}}{N}$$

This is a simple distance metric for two dimensions where C is the candidate sample and T_i is each of the templates in the gesture template set. The path distance for each template is then defined as:

$$score = 1 - \frac{d_i}{1/2\sqrt{size^2 + size^2}}$$

The size value here is the number of points that the candidate samples are normalised to (i.e. 100 points). This value is then used as a score metric. The best value is defined to be the correct gesture. To improve the recognition, the candidates are also rotated using the Golden Section Search to improve the recognition

over the original "guess" that is the indicative angle rotation. This finds the minimum value in a range of values (perfect for finding the local minima i.e. the desired gesture) using the Golden ratio ($\varphi = 0.5(-1 + \sqrt{5})$). The angle for rotating the candidate gesture is incremented by the golden ratio up to 45^o in 2^o increments (based on the paper's results). This is much more efficent that both brute force searching and a hill climbing technique (see Wobbrock et al. [51] for reference).

The limitation of this method is due to the simplicty of the recognition algorithm, some common gestures cannot be differentiated and hence recognised. These are gestures that depend on scale or orientation of the gesture. The other issue is due to the non-linear scaling, objects that are similar in creation, but have variations along, for example, the x axis, but not the y (such as a circle compared to an ellipse, or rectangle compared to a square) cannot be distinguished. Finally time is not taken into account due to the resampling, so the speed at which a gesture is performed is lost.

However, this is a minimal method. It is fast and requires significantly less memory than comparable schemes which is important for mobile devices when runnng at the same time as other processes. The algorithm was implemented in Objective-C on the iPhone. Results of the implementation are shown in the results section of the report.

4.2.5 *Extending to 3-D - The $3 Recognizer*

For 3-D gesture recognition, after comparison of the methods outlined in the related work section, the "$3 recognizer" method was chosen (Kratz and Rohs [21]). This is an extension of the $1 recognizer for three dimensions and is again a minimal and relatively fast algorithm based on template matching. The correct recognition rate is lower than HMM based methods but it is comparable, given the simplicity of implementation and the minimal requriements on the operation system. This is perfect for my application as the requirement is that the recognition should be performed on a mobile device and hence the minimal, and faster solution is prefered (to minimise CPU and memory load hence improving battery life). One negative of such an implementation is that there is no method for automatic segmentation of time (like with more advanced HMM based methods Schlömer et al. [39]) and so there needs to be a button to define the start and end time of gestures.

The steps in the algorithm are extensions of the "$1 recognizer" method (extended to two dimensions) except that the authors propose using a set of templates for each gesture (e.g. 5 circle templates) and a metric to define whether or not the gesture is of a particular type. Due to the complexity of 3-D recognition and the relative simplicity of this method, a naïve threshold gives poor results (and does not make best use of the information). The set of gestures helps to overcome some of the issues of noise and repeatability of the gesture, but at the cost of increased processing load (and hence slower recognition rates). However this set of training data is half the amount of comparable HMM amounts (again good for mobile devices).

The data used was the difference between accleration values, refered to as "acceleration deltas". No preprocessing was proposed in the paper (achieving 80% recogniton rates).

Figure 8: The iPhone as an Accelerometer[5]

The main differences between the $1 and $3 implementation are:

ROTATION: The rotation to the indicative angle which required taking the inverse cosine of the scalar product of the centroid and the start point of the gesture.

$$\theta = \arccos\left(\frac{(p_0 \bullet c)}{\|p_0\| \, \|c\|}\right)$$

The unit vector orthogonal to p_0 and c is then found. This vector is given by the crossproduct of p_0 and c:

$$v_{axis} = \frac{p_0 \times c}{\|p_0 \times c\|}$$

and the rotation is then performed about this vector.

TEMPLATE MATCHING BY FINDING THE OPTIMAL ANGLE: The same scoring metric is used (minimum distance), but the Golden section is extended to two dimensions. The range of values searched is also increased from $\pm 45^{\circ}$ to $\pm 180^{\circ}$ but now in 5° steps, this is much wider search area, but only using a slightly large number of steps. Each template in the library is then searched and a table of the results is produced.

A SCORING HEURISTIC: The table from the template matching step is then sorted, a threshold, ε, is defined and the following heuristic is applied:

- Iff the highest-scoring candidate in the score table has a score $> 1.1\varepsilon$, return this candidate's gesture ID.

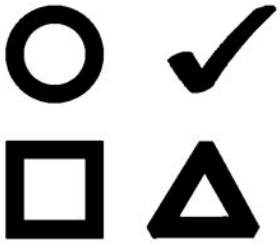


Figure 9: The gesture set

- Iff, within the top three candidates in the score table, two candidates exist of the same gesture class and have a score > 0.95ε, respectively, return the gesture ID of these two candidates.
- Else, return "Gesture not recognized!".

The limitations of such a method are obvious. It is an extremely simple algorithm that does not make use of any noise reduction in the data and hence suffers from the fundamentally noisy data that is accelerometer data. Also having more templates improves the recognition rate, but nothing is learnt from a larger set, simply there are more comparisons, so this method is not scalable to larger gesture data sets. However it is reasonably accurate, the rates for the simplified set are comparable to an HMM based approach (depending on the gesture) and it is fast and simple. Perfect for realtime gesture recognition on a mobile phone (which does not have the computational power required for many of the more advanced HMM based gesture recogntion algorithms).

The results, described fully in the next section are variable, dependant on the type of gesture and the templates recorded for comparison. To the best of my knowledge, this is the first implementation of the $3 recognizer on a mobile device.

4.2.6 *The Gesture set*

In order to keep the recognition step fast, the gesture library was kept small to 4 gestures, however after some testing, both directions for each gesture had to be recorded as people drew the gesture in both directions (and this fooled the template matching). Having small gesture set also reduced the search space for the recognizer, the load on the phone and the number of gestures which needed to be shown to the user. The same shape of gesture was used for both the 2-D and 3-D gesture recognition to reduce the cognitive load on the user. Visual aids as to the nature of the gestures were also given to the user on the visualisation display. The four gestures are a circle, square, triangle (all in both directions) and a tick or check mark (depending on which side of the Atlantic you're from) and they are shown in Figure 9.

The recognition rates of the gesture set are shown in Part III.

4.3 SUMMARY

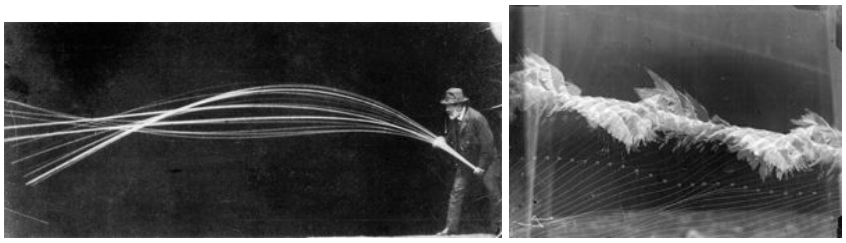
In order to create a natural interface for both navigation and content creation, a mobile phone based controller was developed which performs gesture recognition using the touch screen and accelerometer to record and recognise two dimensional and three dimensional gestures completely on the phone. The methods chosen to perform the recognition were fast, and simple algorithms, rather than more complex Hidden Markov based methods to minimise the computational load on the phone. A small, well defined gesture set was used to further reduce the load as well as keep the novel interface style simple for new users.

In the next section, the visualisation and sonfication of the project will be discussed.

5 VISUALISATION AND SONIFICATION OF THE 3-D ENVIRONMENT

The focus for the system was to analyse gestural recognition as an interaction style for natural user interfaces. Hence any data for visualisation and exploration should come from the user's motion in the tradition of Process art, the 1960s American movement focused on the method of artistic creation rather than the 'object'. By creating shape and form from the user's motion, an interest and affinity with the system is naturally created. This was important to immerse the user, putting them in a relaxing environment which allows them to express themselves; with the end goal being to help analyse the performances of a natural user interface

In this section I will briefly discuss the some of the early time and motion studies and process art as inspiration for the visualisation. Then look into the the technical steps neccesary to create the real time visuals as well as the camera controlled by the navigation controller. Then we will briefly look into the development of stereo vision for use in a visualisation environment such as the Allosphere. Finally we will look into the implications of sound for the natural user interface (a rarely studied area) and sound is created in *Gesture*.



(a) Marey's Chronophotograph of Man with a Pole [27]

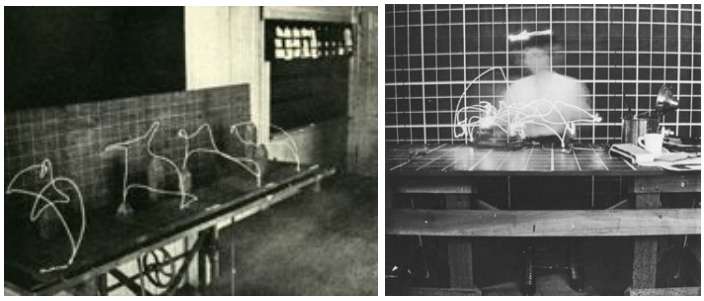
(b) Marey' Capturing a dove in flight[28]

Figure 10: E.J. Marey's Work on Motion

5.1 CAPTURING MOTION FOR VISUALISATION

Some of the first to investigate natural motion in the world were the early chronophotographers. Etienne-Jules Marey was seen by many as the first to make use of timelapse photography to capture the unseen motion of humans and animals (and in many ways was one of the fathers of cinema). Some of his work, shown in Figure 10, helped change people's perception of anatomy, aviation, motion and the study of labour and was said to make "the natural unnatural" .

Frank and Lilian Gilbreth were also interested in labour and focused on 'work simplification'. This was the idea that by analysing workers motions and breaking down the results into fundamental motions (a set they called "therblig" - Gilbreth backwards) you could simplify processes and optimise your workforce. In order to test their theories, they employed chronophotography as a means to record people's mo-



(a) Some of the Gilbreth's Wire Motion Models

(b) Woman at Work - The Gilbreths [15]

Figure 11: Frank and Lilian Gilbreth's Time and Motion Studies

(a) Lynda Benglis - For Carl Andre, 1970[7]

(b) Richard Serra - Lead Piece 1968 [40]

Figure 12: Process Art

tion. Some of the wire models they constructed, shown in Figure 11a, demonstrated for the first time the true path of motion of a worker througha factory. This changed the perspective of factory work for ever and helped to solidify the theories of Scientific Management, the idea that the greatest output of a work force can be achieved by optimising their workflow Taylor [48]. This lead onto the production line and other mass production techniques.

This chronophotography is inspirational to my aims for *Gesture* as the process of capturing motion as a means for expression is an aim for my work as well as the pioneers of the field. Such a method of creation, where the process of creation is the ultimate aim was also crucial to the Process art movement in 1960s America. The focus of the movement was based on the ideal that art should be based on the method of creation rather than the artistic 'object'. Classic examples of this can be seen from Lynda Benglis' poured metal sculptures, where the action of throwing molten lead was the action she saught to capture, or the lead springs created by Richard Serra where the cooling creates new forms (Figure 12). This was one of the main inspirations for the aesthetic style of *Gesture* as the users motion creates the visual form and so their motion should be represented by the final form that is created.

5.2 EXPLORING THE IPHONE DATA

Looking at the raw accelerometer data produced by the iPhone (Figure 13) gives us a form. By then using the data from that motion to inform the colour, thickness and style, then we can create art. Some of the early exploration of these forms are shown in Figure 14. These were focused on the development of the system and so the colour scheme is semi-randomly assigned based on the motion and value of the accelerometer data at that time.

The final visualisation now maps colour (using the HSB scale), thickness of the line (which is actually a sequence of triangular strips) and location based on the integrated accelerometer data (to obtain a displacement vector for each point) shown in Figure 15. By mapping the data in this way, some elements of the

5.3 THE VISUALISATION ENVIRONMENT

The application for visualising the data for *Gesture* was written in the Java based Processing IDE[1]. This meant that visual elements could be rapidly prototyped, the project could run cross-platform and would be easily extendible to include additional features, such as Active stereo or sound creation (both discussed later).

All the graphics were created in OpenGL. OpenGL is a generic software interface for computer graphics hardware meaning OpenGL code will run efficiently on multiple platforms and hardware without you rewriting code. OpenGL defines a set of basic shapes known as geometric primitives (points, triangles and polygons) which are then used to build up a more complex graphics set. The program structure is that of a state machine. It displays one image (and stays in that state) until another set of commands cause the display to update. By sending the sequence of commands to the graphics hardware, the scene is created. Full details of implementation for OpenGL including an excellent introduction to the language and more advanced topics can be found in the OpenGL Red Book [Shreiner et al. [43]].

5.3.1 *Viewing a 3-D scene : Camera control*

In order to view a 3-D scene in OpenGL firstly objects need to be rendered using OpenGL commands (again see Shreiner et al. [43] for full details) and this plots the points in object coordinates. need to be completed so that the desired part of the scene can be viewed. Understanding the sequence that transforms it to window coordinates is important for defining a camera in OpenGL which can then be controlled to view the scene. A camera for computer graphics is analogous to setting up a physical photographic camera in the real world. The modelview matrix is analogous to arranging the scene to be photographed and then setting up the camera to point at it. The projection matrix then defines the lens or the level of zoom of our camera and the shape of the image (if it is skewed etc). The persepective division step normalises the view data so that it can then viewport transformed so it matches the size of the display window (e.g. the monitor). These stages are shown in the steps shown in Figure 16. By changing the projection

1 www.processing.org

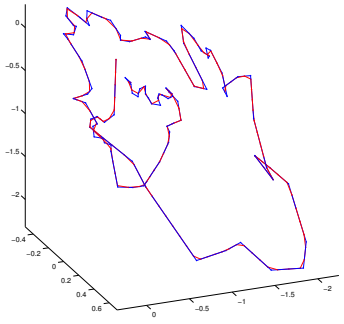
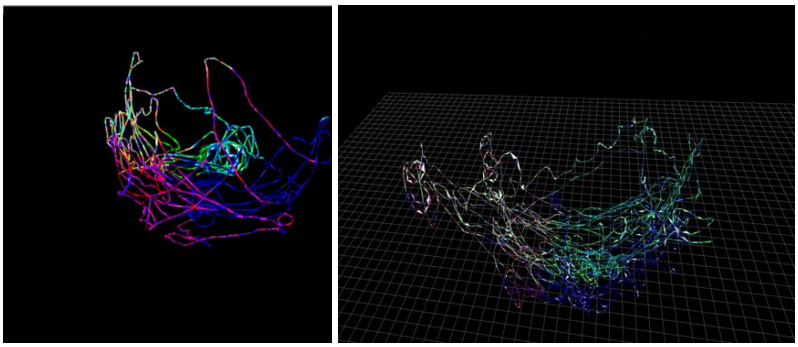


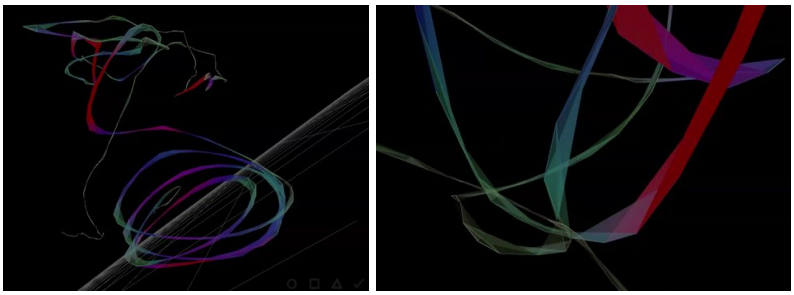
Figure 13: Raw Accelerometer data for a Circle Gesture (Also shown is resampled data for first step of gesture recognition)



(a) An Early Attempt at 3-D visualisation

(b) A later attempt with grid

Figure 14: Early Attempts at Visualising Accelerometer data



(a) Overview of Gesture - Now several visual elements are mapped to the data

(b) Fine Detail

Figure 15: Final Visualisation style

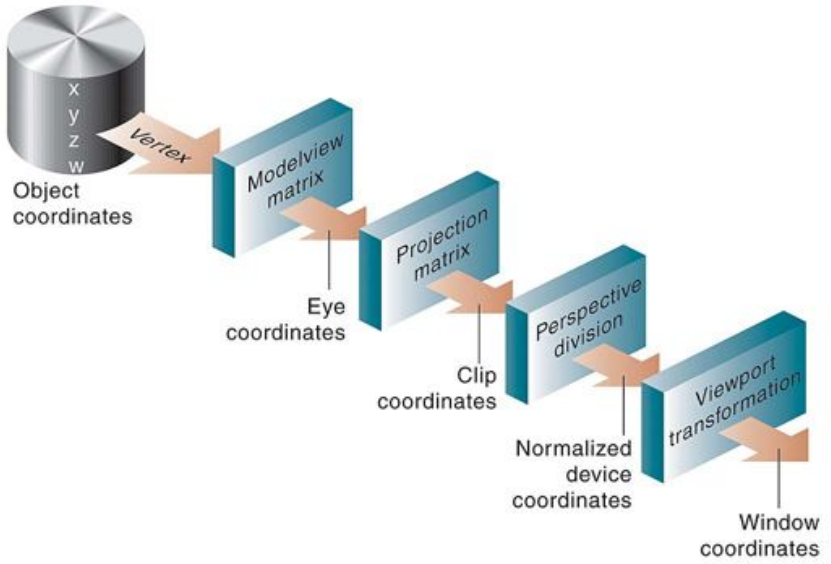


Figure 16: The OpenGL Vertex Transformation [43]

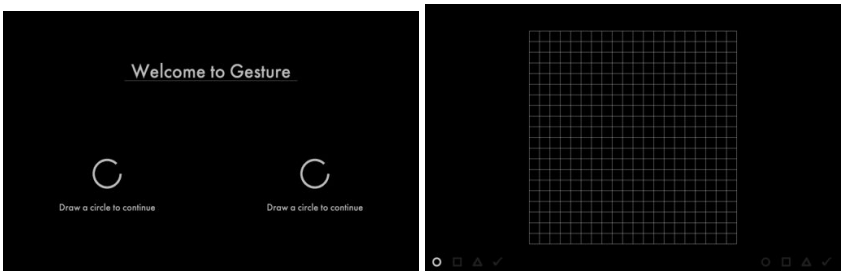
matrix, using standard matrix rotation techniques, addition etc, we can control the zoom level and position of the camera. By abstracting these into function call such as "rotate" etc, we then have a means of controlling the camera from the iPhone. A full tutorial on camera class creation can be found at [4].

5.3.2 *Stereoscopic imagery*

The modern cutting edge visual technique is stereo video. Unlike normal video of three dimensional scenes, stereoscopy provides a separate view for each eye which means viewers can gain a sense of depth from. Since each eye captures a separate image (each eye views the same scene, just separated by a disparity (the distance between the eyes), this needs to be mapped in the computer graphics. The scene is "double buffered" where special graphics cards (which have two visual buffers) progressively update each part of the scene and flip between each part of the scene. In order to ensure that the correct image goes to the correct eye, the user needs to wear special glasses. In the Allosphere this is done using special "active shuttering" glasses. An infra-red signal is sent at rate the projector flips between the two images of the scene (one for the left eye, one for the right) that is produced. The glasses sync to these images and then close each eye successively in time with the image. This is done faster than in perceivable by the eye (using LCDs to black out the eye) and so the correct image goes to the correct eye at the right time.

In *Gesture*, stereo vision was implemented by developing the graphics in true 3-D and then using the stereo-vision library[2] developed by Angus Forbes and Charlie Roberts, two researchers in the AlloSphere group.

2 http://angusforbes.github.com/stereo/



(a) The Gesture "Welcome Screen" (b) The Gesture Viewing Window

Figure 17: Gesture's Visualisation display layouts

5.3.3 *Sonfication*

Earcons are audio notifications. For Gesture, they will be played when gestures are recognised

Sound was generated for the scene using the Minim Library for Processing[3] The key considerations were what the sounds should be, the duration and the differentiating between the earcons and sound generated to complement the visual data. Earcons have been shown to help improve users' understanding of systems as another form of notification[8]. In terms of sonifying the data. In order to increase the likelihood of exploration in *Gesture*, sound is produced when the acceleration is over a certain threshold (so that this reaction is not obvious). The pitch increases as the acceleration increases and when the accelerometer hits the limit, the quantisation of the sound increases and so the distorition of the sounds maps to the distortion of the data (since the accelerometer has an relatively low upperbound beyond which the output is capped. Performing these two sonic cues together helps make *Gesture* a true multi-modal user environment.

5.4 CONSTRUCTING THE UI

In order to maximise the user experience and controll the flow of the user through the program steps, several visual aids and checks were added into the display. Shown in Figure 17b, the bottom left and bottom right of the screen displayed a "Head-Up display" style element which became illuminated when the gesture was recognised. This added to the pre-exisiting signals from the vibration of the phone and the audio "earcons" created by the sound. The grid was added to give perspective to objects in the 3-D scene from which the user could judge depth. This is important as the forms produced are abstract and do not allow for prior perception of the images and so new users cannot gain an estimate of depth from the scene (sometimes even with stereo).

The homescreen was introduced to control access to Gesture and by giving explicit instructions with a visual aid, helping the user by providing clues as to how they should interact with the system. This was so successful that people attempted to draw on the projected wall when they did not realise this required the iPhones as controllers. A second page with text was also displayed that the user then had to read and draw a gesture to confirm having read. This provided information about the system and clues as to how to interact with it.

The relationship between the gestures and the action of the visualisation was also of crucial importance. With such a limited gesture set,

3 http://code.compartmental.net/tools/minim/

	DRAW	NAVIGATE
Circle	Cycle through the colour schemes	Auto rotate the camera : on/off
Square	Cycle through fill options	Take the camera to predefined location 1
Triangle	Cycle through line plotting options (thickness etc)	Take the camera to predefined location 2
Check	Begin/end drawing (in addition to explicit button)	Take the camera to predefined location 3

Table 2: What each action does in *Gesture* N.B. The mapping was the same for both 2-D and 3-D gestures

making a good selection of gesture for a given action is hard. A good guide from previous interface designs listed in Chapter 2 is to assign a metaphor to the gestures. This helps the user to recall the actions and creates a better mental model of the system for the user. Hence that is why for the Navigate app, the circle gesture can easily be assigned to switching on and off the camera, but then what should the triangle represent. Often, an arbitrary gesture should not matter as long as the system is responsive and the behaviour is consistent, which was a key aim for the system.

Once this was completed and the mappings were completed, then the system could be tested with real users. In the next section, an evaluation of the accuracy of the gestures is performed as well as the users' performance and their satisfaction with the system and thoughts on the new interaction style are presented. Finally the overall lessons learnt from this project are discussed.

6 CONNECTING AND FINISHING THE SYSTEM

The iPhone controller and the visualisation needed to be connected in order to work properly. The iPhone has a wireless card which means data transfer can be done over the wireless network. However in order to do this, a connection protocol needs to be established that is fast and efficient as well as easily understandable. In this section we briefly discuss networking protocols and messaging and how the connections are then used in *Gesture*.

6.1 BUILDING A CONNECTION - UDP VS. TCP

Using the built-in wireless capabilities of the iPhone makes transmission of data between the phones and the computer handling the visualisation easier. This means we only have to worry about the organising the transport layer of the system. The transport layer is a group of protocols responsible for taking data from the application layer (i.e. your program) and converting it into data packets which can then be transmitted over the network (through the network layer). This is all defined by the Open System Interconnection model (or OSI [35]) There are various protocols for doing this, two of the most commonly used examples of which are UDP (User Datagram Protocol) and TCP (Transmission Control Protocol) :

TRANSMISSION CONTROL PROTOCOL (TCP): TCP is one of the most common protocols for computer to computer connections. The main reason is that TCP provides guaranteed delivery of data packets in the correct order and provides error correction. This is because TCP implements flow control and so checks to see if each packet has been successfully transmitted. If not, then the system will re-transmit the packet to ensure delivery. The issue is that this can introduce a delay in getting this consistent data.

USER DATAGRAM PROTOCOL (UDP): UDP is a connectionless, broadcast protocol. This means it sends the data and does not care whether it has been successful or not. This means it can be extremely unreliable with packets being dropped, sent out of order or even sent twice. This has to be handled by the application (if this will cause errors) but it means that there is next to no overhead for UDP transfer meaning it is extremely fast. UDP is used for streaming video and audio, but this is another reason why the quality of this kind of media can be low.

Hence for *Gesture*, UDP is a good choice as the data for rotation and drawing requires speed to be effective and create a natural environment. The issue of packets being dropped or out of order can impact performance, and impact the smoothness of the camera motions. In an improved version of the system, TCP and UDP could be used to separate the information out. Gestures, which require guaranteed transmission should be sent via TCP, but as we are using a mobile device and battery life is a challenge (continuous broadcast drains the battery), UDP is the best solution.

6.1.1 *Organising your data - OpenSoundControl*

UDP does not organise your data, it leaves the organisation of the information to the Application. Hence we need a data encapsulation format which can be easily written and easily read, as well as being minimal for transfer. OpenSoundControl, a message based protocol, was developed for easy transmission between mobile devices. The full standard can be found here [52]. For the project the minimal implementation liblo[1] was used for objective-C and oscP5[2] for Processing. This also included support for the creation of the necessary UDP connection.

6.2 COMPLETING THE SYSTEM

Gesture makes use of a wide range of components and technologies to attempt to build a system which has a natural user interface. The iPhone controller UI layout as well as methods for gesture recongition have been discussed. Then the aspects of visualisation in OpenGL were covered including the porting to Active Stereo and to include sonification and earcons for the data. Finally the connection of the controller and the visualisation was discussed. In the next section we will analyse the results of this work and look at what conclusions can be drawn from this.

1 http://liblo.sourceforge.net/
2 http://www.sojamo.de/libraries/oscP5/

Part III

USER TESTING, RESULTS AND DISCUSSION

7 USER TESTING

There were two things that needed to be user tested in order to make a critical evaluation of the system. The first is the effectiveness of the gesture recogntion techniques and the second is the effectiveness of the overall system including a critical evaluation. Ten participants were used for the study of the gesture recognition. The same users were used for both tests, since it was assumed that these were separate tasks. Quantative results are presented. In order to measure, the effectiveness of the overall system, ince the end task does not have a measureable outcome which we can use as a comparison (a weakness of the system), the results focus on subjective user evaluation. The results breakdown is presented in Appendix A

7.1 GESTURE RECOGNITION

7.1.1 *The participants*

The ten users in the trial were selected to cover a range of abilities and both genders. Five men and five women were selected. Several things were recorded such as their right or left handed-ness, prior experience with touchscreen or accelerometer based applications (all had used iPhones or played Wii before) and user success at achieving the desired gesture.

In both cases templates were pre-loaded, unlike the testing featured in papers such as Kratz and Rohs [21] and Schlömer et al. [39]. These studies trained their recognition algorithms to each user. This testing, however, was done blind. As such the results present a good result, although the variance of the 3-D gestures is extremely high, highlighting the fact that everyone performes gestures in 3-D differently and so building a generic algorithm to solve this issue is extremely challenging.

7.1.2 *Touch Screen gestures*

The results speak for themselves. This is an extremely accurate gesture recognition method. The limitation is that the gestures themselves have to be learned and completed exactly. Lazy completion of the gesture or moving too fast were the main reasons for the failed recognition. The one issue however was that people draw the gestures in different directions (clockwise or anticlockwise). This was not a problem for the circles or triangles (which had templates for both directions) but the square only worked in one direction. Interestingly, even the check was performed the opposite direction by a left handed person.

Most of the feedback was positive with users feeling that they could perform the gestures accurately and saw tangible applications for such a system in their own lives. Some users commented that they felt that the limitations gestures were more down to their own failings (found physical tasks challenging - one user had trauma dexterity) or felt their memories and interaction style was more aural or visual rather than spatial.

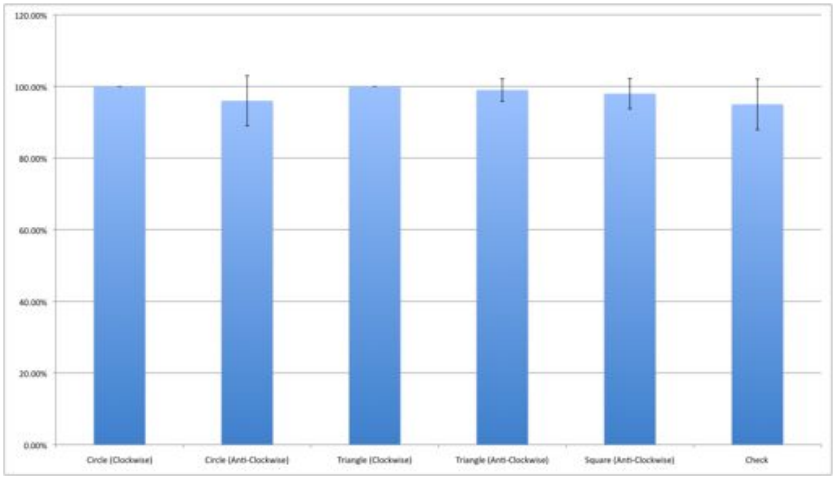


Figure 18: Results for Touchscreen based Gesture Recognition

7.1.3 *Three-Dimensional Dynamic gestures*

where templates were recorded for each individual user and then the effectiveness of the algorithm was tested, this testing was done as a final complete system solution. The basic templates chosen were from a range of users and then the effectiveness of the system new users to use this system was tested. Five users (not part of the ten surveryed) were recorded each doing the gesture five times. One template from each was selected at random to make a 5 template set for the recognition folder.

As can be seen from the results, the gesture recognition is highly dependent on the proximity of the template to the gesture performed. For example the triangle gesture results ranged from 0% to 100%. This was down to poor templates for the triangles (whilst most gesture templates clustered together in the ranking table, triangles were often dispersed) and some users were extremely poor at being able to draw repeatable gestures or even close to other users (for exampler user 5 averaged less than 10%). This obviously presents problem for the construction of a system using 3-D gestures (the false positive level is also too high) and so more work must be done to improve this area to make it useful for practical applications.

The feedback received was extremely positive. Most users said it was more "fun" or "cooler" to interact with a computer in this way, as can be shown with the success of the Wii versus the XBox. Again the same issues cropped up where towards the tenth gesture, users focused less on performing the gesture accurately and so it would not be recongised. Most would have liked such a means of interaction for use around the house or to control their music etc, but only a few saw this a real practical application they could use with their existing computers.

7.2 THE GESTURE SYSTEM

The *Gesture* system was positively recieved at both public demonstrations. Some of the problems of releasing it "in the wild" were that most users were unsure of how to use gestures. Many people drew circles as one and a half circles rather than a complete single circle (twelve o'clock till twelve o'clock) which was unexpected and not designed for.

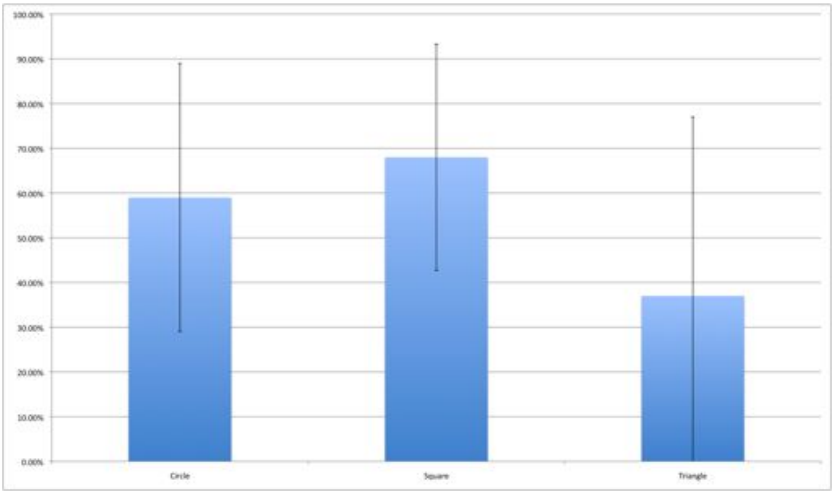


Figure 19: Results for 3-D Gesture Recognition

The solution to this is to guide the users as the draw a circle command did. In fact this was so successful in guiding peoples actions that many people tried to draw a circle on the projected wall (without realising they needed the iPhone controller!). Improvements would be adding guides for each gesture, or running each user through a training section at the start.

The main issues with the 3-D gestures was that everyone draws shapes in 3-D space in their own way. The templates did not capture the range of gestures people performed. In fact, to fully solve the template matching approach a number of gestures required would be larger than most computer's memories, let alone that of a mobiles phone.

The other major issue that was found when running the gesture recognition at the same time as sending data for drawing or rotating the camera. When the processor was handling both operations, the polling rate of the accelerometer was slower and so the number of sample points for each gesture was lower. This missed key details of the gesture and hence reduced the accuracy. The simple improvement that was made was that sending is stopped temporarily whilst the "Gesture" button is pressed.

At the second demonstration to more technical users, the ability to pick up the gesture language was much faster. Many of the comments were very positive. The reception of the Navigate app was also extremely positive with JoAnn Kuchera-Morin, creator of the AlloSphere, saying that she needed one for the Allosphere.

The main discovery from the demomstrations was the huge range of user expectations. Many found the mappings strange, but then just a many found them instantly intuitive. This suggests that the ultimate solution should simply be fully customisable where you can retrain the gestures for yourself.

8 CONCLUSION

Gesture was created to build a user interface which demonstrates the potential of new methods of Human Computer interaction. This is a neccesary direction as the paradigm of the keyboard and mouse starts to break down in the age of mobile computing that we now live in. We need new, more natural ways to interact with our computers on our terms and not by being constrained to the desk and chair.

The place of the keyboard and mouse is the office and in that environment it reigns supreme, but as more computing power moves from the desk to the mobile phone or the television, the suitablity of newer ways to interact must be investigated.

Gesture recognition is a relatively new means to interact with computers, and along with speech recognition, represents a key stage in making a natural interface. By implementing gesture recognition for both the iPhone's touchscreen and the more experimental method using the accelerometer, the potential of this technology can be investigated properly.

The findings of this work are that gesture recogntion using touchscreens is now a mature enough technology that it is ready for implementation in commerical products. The iPhone has shown that basic gesturing has a place in mobile operating systems and my results show that more complex gestures can be easily recognised and recognised accurately. The problem is context.

In order to make gesture recognition realistic, it needs to be tied to a strong context or metaphor which makes it appropriate. Drawing a circle to plot a triangle on the screen is an obvious exaguration of bad design, but ensuring that the gesture is appropriate is a huge challenge for the interface designer. As we have seen from the results of this work, even simple geometric gestures represent a challenge when a user is told "draw a circle" and the designer has to handle all of the resulting scenarios. Providing a visual guide or strongly tying the action to a metaphor reduces the ambiguity to the user and so narrows the resulting inputs to a more manageable number. This was done in *Gesture* at the home screen, but could have been used more widely for these reasons.

This question of context is crucial for the development of a natural interface. Humans do not use the same method to interact with each other all the time. Sometimes we whisper, sometimes we shout, sometimes we gesture and sometimes we write. Similarly, voice recognition would be great for controlling a "smart" home, but would be a terrible method for control on a plane or to enter commercially sensitive information (as you would have to speak it out loud). Again the challenge as we move towards a more pervasive style of computing is do we develop the interface to be suitable to all environments or should we contextualise it for the user making it a specialist with its own functional environement.

Gesture created a context for gesture recognition where it fitted in and worked effectively. Controlling three dimensional visualisations is a particularly challenging area for Human Computer interaction as intuitively mapping user input from a two dimensional controller

to three dimensions is hard. Adding into the that the fact that what one person finds extremely intuitive, another person finds completely alien and challenging and providing a generic solution is probably impossible.

Customisation of the interface is a crucial step is overcoming the uniqueness of users and one of the key insights to making a true natural interface. If a user finds the interface strange, then they should be allowed to change it to suit their needs. Many of the people who tested gesture or used the gesture recongition found it interesting and useful, but not the way that works best for them. Humans have many learning styles and hence many ways of interacting, be it verbal, physical or simply writing it down, and the Natural User Interface has to cater for all of these people.

Making *Gesture* a multi-user system also allowed another key challenge of a natural user interface to be investigated. As realtime communication becomes more pervasive and systems use the power of the internet to connect more and more people together simultatiously, designers need to cater for multiple users. Beyond the technical challenges of data handling etc, there is also the issue of creating an environment in which it makes sense, is engaging and works in realtime. By separating the navigation and draw functions into two separate controllers, both could become more specialised and provide more functionality to the user and by working together the display could become more engaging and interesting.

The main comment from users of *Gesture* is that is was fun to use. If "natural" user interfaces are challenging and require a steep learning curve to understand, not only will they no longer be natural, but they will not be taken up by users, even if it would be beneficial in the long run. Employees work more effectively when they are in an inspirational and fun environment and it is just the same with computer users. If the system is rewarding, responsive and engaging, then the barriers to the raw power can be broken down and the true potential of computers can be realised.

Creating *Gesture* has been a challenging and informative experience. I have learnt a lot about gesture recognition, interface design and wireless protocols, but ultimately the future of the natural user interface is about people. In order to make systems for new environments (or ones in which computers have not classically be found in) that are easy to use, we need to design for the responses of people in these situations and build the system that caters for their desires and needs.

Part IV

APPENDIX

A

RESULTS - RAW DATA

The table in Figure 20 presents the raw data of the user testing for accuracy of the implemented gesture recognition algorithms.

A.1 OPEN RESPONSE - USER FEEDBACK

- "2D was easier to use than 3D.3D was difficult to visualise, so was harder to repeat the same shape and size of drawing. It was difficult to make the beginning and end point meet. However, I enjoyed using it and I look forward to using it in the future."
- "The drawing was easier to control. Making the same start and end point was hard, but I loved using the 3D gestures more. It was much more interesting, but less repeatable and I guess more challenging to get it to work. Personally I find it hard to visualise shapes, so gestures aren't probably the best thing for me."
- 3D was a better mapping for the screen. Using 2-D movements limited the screen as your fingers get in the way. I would have liked to have been able to have seen what I'd drawn for both. but in general the 2D didn't feel as natural. 2D didn't have proper affordance. For the 2D I found myself getting distracted. The 3-D allowed more interaction but I feel like I needed to be more precise. Having some sort of tactile affordance would improve the naturalness. I can see though that in a closed environment 2-D would be more practical."
- "Very cool and very intuitive. Being exact with the shapes is easy. Was much more fun as I really liked the interactitvity, kinda like on the wii!"
- " I found this really hard. Maybe having a training period at the beginning of the application. I sometimes forgot the direction of movement I had done. Maybe by having audio cues within the gesture, such as when you hit the three points of the triangles would help, especially if it got stronger as I got closer to making the real gesture. Similarly visually seeing this as well would help!"
- "This is awesome. Sadly i find this stuff quite hard as I've been diagnosed with trauma dexterity and I guess being left handed too didn't help. Nice work though."
- "I definitely found this to be a more natural way of doing stuff. 3-D stuff gave you more control that 2D but definitely better to have both. I think triangles were hard in 3D and were definitely more natural on paper. But yeah, when this gets released on the App Store let me know"
- "The controls were totally non-intuitive, Could have made more of the gesture stuff."
- "I thought the controls were very intuitive. I was really impressed with the navigation most. We need one of these in the AlloSphere"

- "This is great. It really reminds me of Miro's wireframe models. I enjoyed using it a lot. Well done."

2D

User	1	2	3	4	5	6	7	8	9	10	Average	Std Dev
Circle (Clockwise)	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100.00%	0.00
Circle (Anti-Clockwise)	100%	100%	80%	100%	100%	90%	90%	100%	100%	100%	96.00%	0.07
Triangle (Clockwise)	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100.00%	0.00
Triangle (Anti-Clockwise)	100%	100%	100%	100%	100%	100%	100%	100%	90%	100%	99.00%	0.03
Square (Anti-Clockwise)	90%	100%	100%	100%	100%	90%	100%	100%	100%	100%	98.00%	0.04
Check	80%	100%	100%	100%	100%	90%	90%	90%	90%	100%	95.00%	0.07

3D

User	1	2	3	4	5	6	7	8	9	10	Average	Std Dev
Circle	80%	40%	90%	60%	0%	40%	100%	40%	60%	80%	59.00%	0.2998
Square	60%	60%	100%	60%	20%	80%	100%	80%	80%	40%	68.00%	0.253
Triangle	10%	0%	100%	0%	0%	80%	40%	80%	60%	0%	37.00%	0.4001

Figure 20: Raw results

BIBLIOGRAPHY

[1] URL http://emilyethos.files.wordpress.com/2009/09/brain-computer-interface-11.jpg. (Cited on page 10.)

[2] URL http://www.technovelgy.com/graphics/content08/contact-lens-circuit-human.jpg. (Cited on page 10.)

[3] URL http://paloma.isr.uc.pt/gesture-reco/pics/gestureLib.jpg. (Cited on page 10.)

[4] URL http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=Quaternion_Camera_Class. (Cited on page 35.)

[5] Apple. iphone accelerometer picture. URL http://developer.apple.com/iphone/library/documentation/uikit/reference/UIAcceleration_Class/Art/device_axes.jpg. (Cited on pages ix and 27.)

[6] David F. Armstrong, William C. Stokoe, and Sherman Wilcox. *Gesture and the nature of language*. Cambridge University Press, 1995. (Cited on page 22.)

[7] Lynda Benglis, 1970. URL http://www.brooklynmuseum.org/eascfa/feminist_art_base/archive/images/576.1721.jpg. (Cited on page 32.)

[8] Meera M. Blattner, Denise A. Sumikawa, and Robert M. Greenberg. Earcons and icons: Their structure and common design principles. In *Human-Computer Interaction*, volume 4, pages 11–44, 1989. (Cited on page 36.)

[9] Klaus Boehm, Wolfgang Broll, and Michael A. Sokolewicz. Dynamic gesture recognition using neural networks: a fundament for advanced interaction construction. Technical report, Proc. SPIE, 2004. (Cited on page 24.)

[10] Richard A. Bolt. "put-that-there": Voice and gesture at the graphics interface. Technical report, Architecture Machine Group, Massachusetts Institute of Technology, 1980. (Cited on page 17.)

[11] Mahmoud Elmezain, Ayoub Al-Hamadi, Jorg Appenrodt, and Bernd Michaelis. A hidden markov model-based continuous gesture recognition system for hand motion trajectory. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, 2008. (Cited on page 24.)

[12] D. M. Gavrila. The visual analysis of human movement: A survey. In *Computer Vision and Image Understanding*, volume 73, pages 82–98, 1999. (Cited on page 24.)

[13] Ron George and Joshua Blake. Objects, containers, gestures, and manipulations: Universal foundational metaphors of natural user interfaces. In *CHI'10 Natural User Interfaces Workshop*, 2010. (Cited on pages 3 and 9.)

[14] DIETER GERNERT. Ockham's razor and its improper use. URL http://www.vixra.org/pdf/0908.0074v1.pdf. (Cited on page 9.)

[15] Frank Gilbreth. Motion efficiency study, 1914. URL http://click.si.edu/images/upload/Images/pn_737_Image_167.jpg. (Cited on page 32.)

[16] Chris Harrison, Desney Tan, and Dan Morris. Skinput: Appropriating the body as an input surface. In *CHI 2010*. ACM, 2010. (Cited on page 10.)

[17] Tobias Hollerer, JoAnn Kuchera-Morin, and Xavier Amatriain. The allosphere: a large-scale immersive surround-view instrument. In *Emerging displays technologies: images and beyond: the future of displays and interacton*. ACM, 2007. (Cited on page 9.)

[18] Juha Kela, Panu Korpipa, Jani Mantyjarvi, Sanna Kallio, Giuseppe Savino, and Luca Jozzo Sergio Di Marca. Accelerometer-based gesture control for a design environment. In *Personal and Ubiquitous Computing*, 2005. (Cited on pages 17 and 24.)

[19] Adam Kendon. *Gesture: visible action as utterance*. Cambridge University Press, 2004. (Cited on pages 4 and 22.)

[20] Nils Krahnstoever, Sanszhar Kettebekov, S. Kettebekov, Rajeev Sharma, Mohammed Yeasin, and R. Sharma. A real-time framework for natural multimodal interaction with large screen displays. In *Fourth IEEE International Conference on Multimodal Interfaces*, 2002. (Cited on page 17.)

[21] Sven Kratz and Michael Rohs. A three dollar gesture recognizer : Simple gesture recognition for devices equipped with 3d acceleration sensors. In *Intelligent User Interfaces*, February 2010. (Cited on pages 24, 26, and 43.)

[22] David M. Krum, Olugbenga Omoteso, William Ribarsky, Thad Starner, and Larry F. Hodges. Speech and gesture multimodal control of a whole earth 3d visualization environment. In *Proceedings of the symposium on Data Visualisation*, 2002. (Cited on page 17.)

[23] James A. Landay and Brad A. Myers. Interactive sketching for the early stages of user interface design. 1995. (Cited on page 17.)

[24] Jiayang Liu, Zhen Wang, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uwave: Accelerometer-based personalized gesture recognition. Technical report, Rice University and Motorola Labs, June 2008. (Cited on page 24.)

[25] E. Mandler, R. Oed., and W. Doster. Experiments in on-line script recognition. In *4th Scandinavian Conference of Image Analysis*, pages 75–86, 1985. (Cited on page 23.)

[26] Jani Mäntyjärvi, Juha Kela, Panu Korpipää, and Sanna Kallio. Enabling fast and effortless customisation in accelerometer based gesture interaction. In *MUM*, 2004. (Cited on page 24.)

[27] Etienne-Jules Marey. Man with pole. URL http://content.stamen.com/talks/where_20_2008/files/marey_pole.jpg. (Cited on page 31.)

[28] Étienne-Jules Marey. Doves, Around 1882. URL http://www.probehead.com/log/images/20050130_Marey.jpg. (Cited on page 31.)

[29] David McNeill. *Hand and Mind: What Gestures Reveal about Thought*. University Of Chicago Press, 1996. (Cited on page 22.)

[30] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. In *Computer Vision and Image Understanding*, volume 81, pages 231–268, 2001. (Cited on page 24.)

[31] Kouichi Murakami and Hitomi Taguchi. Gesture recognition using recurrent neural networks. Technical report, roceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology, 1991. (Cited on page 24.)

[32] Gerrit Niezen and Gerhard P. Hancke. Gesture recognition as ubiquitous input for mobile phones. In *UbiComp '08 Workshop W1 – Devices that Alter Perception*, 2008. (Cited on page 24.)

[33] Donald Norman. *The Design of Everyday Things*. Doubleday Business, 1990. (Cited on pages 4 and 7.)

[34] Antti Nurminen and Antti Oulasvirta. 10 designing interactions for navigation in 3d mobile maps. In *Lecture Notes in Geoinformation and Cartography*. Springer Berlin Heidelberg, 2008. (Cited on page 9.)

[35] David M. Piscitello and A. Lyman-Chapin. *Open Systems Networking: TCP/IP and OSI*. Addison-Wesley, Boston, MA, USA, 1st edition, 1993. (Cited on page 39.)

[36] Réjean Plamondon and Sargur N. Srihari. On-line and off-line handwriting recognition: A comprehensive survery. In *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, volume 22, pages 63–84, 2000. (Cited on page 23.)

[37] Roope Raisamo. Multimodal human-computer interaction : a constructive and empirical study. Master's thesis, University of Tampere, December 1999. (Cited on page 3.)

[38] Byron Reeves and Clifford Nass. *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*. Center for the Study of Language and Information, 2003. (Cited on page 4.)

[39] Thomas Schlömer, Benjamin Poppinga, Niels Henze, and Susanne Boll. Gesture recognition with a wii controller. In *Tangible and embedded interaction*. University of Oldenburg, 2008. (Cited on pages 24, 26, and 43.)

[40] Richard Serra. Lead piece, 1968. URL http://www.kunstmuseumsg.ch/. (Cited on page 32.)

[41] Ben Shneiderman. Direct manipulation: A step beyond programming languages. In *joint conference on Easier and more productive use of computer systems. (Part - II): Human interface and the user interface*, 1981. (Cited on page 4.)

[42] Ben Shneiderman. *Designing the User Interface*. Addison Wesley, 1997. (Cited on page 7.)

[43] Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis. *OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R)*. Addison-Wesley, August 2007. (Cited on pages ix, 33, and 35.)

[44] Sreeram Sreedharan, Edmund S. Zurita, and Beryl Plimmer. 3d input for 3d worlds. In *OzCHI*, 2007. (Cited on page 24.)

[45] Ivan Sutherland. Sketchpad: A man-machine graphical communication system. Technical report, Massachusetts Institute of Technology, 1963. (Cited on page 5.)

[46] Desney S. Tan, George G. Robertson, and Mary Czerwinski. Exploring 3d navigation: Combining speed-coupled flying with orbiting. In *SIGCHI conference on Human factors in computing systems*, pages 418 – 425, 2001. (Cited on page 9.)

[47] Charles C. Tappert, Ching Y. Suen, and Toru Wakahara. The state of the art in on-line handwriting recognition. In *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, volume 12, 1990. (Cited on page 23.)

[48] Frederick Winslow Taylor. *The principles of scientific management*. 1st World Library - Literary Society, 1923 (Reprint 2005). (Cited on page 32.)

[49] Tamas Vajk, Paul Coulton, Will Bamford, and Reuben Edwards. Using a mobile phone as a "wii-like" controller for playing games on a large public display. In *International Journal of Computer Games Technology*, 2008. (Cited on page 24.)

[50] Jamie A. Ward, Paul Lukowicz, and Gerhard Troster. Gesture spotting using wrist worn microphone and 3-axis accelerometer. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, pages 99 – 104, 2005. (Cited on page 24.)

[51] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: A one dollar recognizer for user interface prototypes. In *User Interface Software and Technology*, October 2007. (Cited on pages ix, 23, 24, 25, and 26.)

[52] Matt Wright. The open sound control 1.0 specification. URL `http://opensoundcontrol.org/spec-1_0`. (Cited on page 40.)

[53] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: an interface for sketching 3d scenes. In *International Conference on Computer Graphics and Interactive Techniques*, 2006. (Cited on page 17.)