MASSEY UNIVERSITY

A Near Touch User Interface for Touch Screen Based Systems

A thesis presented in partial fulfilment of the requirements for the degree of

Masters of Engineering

in

Electronics and Computer Systems Engineering

at Massey University, Palmerston North,

New Zealand

Nicholas John Taylor

2012

Abstract

Industry has been heavily pushing for new methods of human-computer interaction in the last several years and this has seen many different technologies move into the mainstream, from infrared sensors[1] and z-cameras[2] to touch screens[3]. Because touch screens are a more mature and developed technology they provide an ideal platform for mainstream technology development, but their level of interactiveness is limited, and these limitations must be overcome or compensated for with clever interface design.

In this thesis, a solution to these inherent limitations in touch screen interface design is proposed by augmenting the touch screens interaction capabilities with one or two cameras to enable a near touch user experience on top of the standard touch screen. This offers flexibility in system design (the near touch is implemented as an extra layer, and can be activated only if present) as well as providing an inexpensive solution. Several Image processing algorithms relevant to this task are also discussed and their implementation evaluated.

Preface

The aim of this thesis is to demonstrate that a suitable near touch interface can be created to allow the cost effective augmentation of basic touch screens to further enhance their practicality and usability as a next generation human-computer interface.

Chapter 1 begins by describing the current issues with human-computer interfaces on touch screen devices, and why they are insufficient to replace the current standard of mouse and keyboard in a conventional desktop environment. This will be followed up by a description of the proposed solution to this issue, and an assessment of its advantages and disadvantages within the context of its intended application.

Chapter 2 will follow on by describing existing systems for human-computer interaction with touch screen devices, as well as non-touch based interaction methods, and list their strengths and flaws in relation to how they relate to the stated problem.

Chapter 3 will cover the design and implementation of this system, discussing relevant techniques, and then assess the quantitative and qualitative results gained. Finally, chapter 4 will describe the strengths and weaknesses that the system demonstrated, and what areas could be improved upon, as well as future development.

The project was carried out in conjunction with our industry partner

Unlimited Realities[1] under a Foundation of Research and Technology TIFF fellowship, and was designed to be integrated into their product Umajin™ [2], and enabled in any supporting Fingertapps™ applications built. It was also intended to create a set of image processing functions that could be implemented in any Umajin™ project on any image or video object.

[1]Unlimited Realities, PO Box 8015, 60 Princess Street, Palmerston North, 4446, New Zealand. Phn: +64 6 3564120, Website: www.ur.co.nz

[2]Umajin™ is a platform that allows applications to be rapidly developed and provides a wide range of functionality and diversity to be supported via integrated functions within the application programming interface (API). It is a commercially available product, and has a specialised branch called Fingertapps™ which is dedicated to the creation of touch based applications, and has support for multi-touch environments.

Acknowledgements

I would like to thank my supervisors Professor Hans Guesgen and Dr. Amal Punchihewa, as well as Mr. David Brebner of Unlimited Realities for their invaluable support, advice and assistance throughout the course of my masters project. I would also like to acknowledge the involvement and support of Unlimited Realities as the Industry partners of this project as part of the Foundation of Research and Technology TIFF fellowship program and thank them for giving me the opportunity to work with them to achieve this result. Finally I would like to acknowledge Mr. Paul Lyons for his thesis writing guide, and the assistance it gave me in preparing this document.

Contents

List of Figures

Chapter 1

Introduction

With the influx of touch based devices to the consumer market, touch based user interaction has become an important and relevant area of study. Although it is a popular interface between user and computer, it is not without its flaws. This chapter will cover the perceived problem with current touch screen based solutions in the area of interaction and interface design, and propose a solution to this issue.

1.1 Problem

The popularity of touch screen technology has surged in the new millennium due to media influences like the 2002 movie Minority Report showing spectacular visions of a future where computers can be interacted with by merely waving your hand in front of them, and where objects on screens can be selected touching the object of your desires in a simple and highly intuitive digital system, allowing even highly inexperienced users to effortlessly use these systems to the fullest of their capabilities without the need for training or to read complex and seemingly obfuscated manuals.

With devices such as the iPhone™[4] and iPod™ Touch [5] and the iPad™[6],

the Microsoft Surface^{TM}[7] and Google Android^{TM}[8] ect., as well as support for touch screens and tablet PC's being a high priority for Windows^{TM} 7[9] and even more so with the new Metro interface central to Windows^{TM}8[10], it is evident that industry puts touch screens on a lofty pedestal as the way of the future. With this mass proliferation of touch devices it becomes important for their marketers and designers to develop simple and effective interfaces to allow them to be used.

This presents a challenge to user interface designers, as the touch screen will only allow for $2 * n$ degrees of freedom where n is the number of individual touches trackable in the x/y directions (improved to $3 * n$ if the touch screen has pressure sensitivity). Conversely a mouse and keyboard interface has the vast potential of 102 keys (standard US keyboard layout), 7 mouse buttons or more, 2 axis the mouse button may be moved on, and a bi-directional scroll wheel with multiple divisions of movement (sometimes multiple scroll buttons are even present). While devices like the iPhone^{TM} have proven popular with the consumer and simple to use and practicality in a mobile environment, their ability to be used to perform complex tasks is limited, and not suitable for a professional or home desktop environment.

Touch screens offer a major innovation that makes them desirable in modern computer systems. They do not require any external input devices to operate (with the occasional exception of a stylus type pointing device, which is often compact and easily stored with the device). This is a large advantage for portable devices such as smart phones and handheld game consoles etc. as it reduces space requirements and weight. This also is useful for items such as public information or advertising kiosks where theft or damage from misuse can occur.

Another advantage touch screens possess over traditional computer systems is that they create an interactive experience for the user that is more engaging and interesting[11] than using the standard mouse and keyboard interface that has become common, and may even be mentally associated with work. Increasing user engagement is an especially important for advertising kiosks.

Implementing the variety of interaction of a mouse and keyboard interface is a large challenge for a touch screen interface as there is only one method of interaction, the touch, as opposed to the mouse and keyboards ability to type with any one of a large number of keys, many of which have become customised to specific tasks such as launching a media player, skipping tracks and pausing/stopping play etc., through to the mouses precise ability to move a cursor anywhere, and click on objects using nine or more different buttons [12]. These two different systems offer extremely different levels of flexibility, with cursor hovering, and the combinations of over one hundred keys that can be pressed in conjunction with several mouse buttons contrasting with only a single (or in the case of more expensive systems, multiple) touch.

To make up for the lack of flexibility touch based systems must use gestures to increase the depth of interaction, but using more than two or three touch points can lead to overly complicated gestures that are impractical for the user. Another issue with touch systems is that it is less precise than the pixel accurate pointer of a mouse, as there is generally a large contact point for the users finger, and it is unreasonable to expect the user to determine the exact centroid of their touch for accurate use, especially when their finger covers the object they are trying to accurately touch (although this is largely mitigated with a stylus system) and may roll or move as it is a malleable rounded surface in contact with

a flat one.

The simplest and most frequent implementation of such touch system is to use the touch interaction as being analogous to a mouse click. The lack of keyboard availability is usually overcome by the use of an on screen keyboard (in Windows^{TM} this is usually a user accessibility feature). This can be a highly impractical solution as it takes up large amounts of screen space at a size practical for many touch screen devices. As a result of this a normal keyboard is usually used when available and practical. This still leaves the issue of only having a touch interaction instead of a full cursor, resulting in the requirements for interface design being much stricter than they would otherwise be.

1.2 Proposed Solution

With the problem of there being insufficient degrees of freedom in touchscreens to perform complex tasks within desktop environments explored, the following section will describe the proposed solution to these issues.

1.2.1 Overview

The solution I propose to the problem of user interface design constraints on traditional touch screen based systems is to create an extra dimension of interaction with the system, by implementing a layer of functionality similar to having a mouse cursor present. Adding this extra dimension of interactivity will allow the user to perform actions analogous to holding a mouse cursor over an object for more information or navigating a branching menu seamlessly without the need for clicks to open new branches, as is possible with current mouse based systems. This new dimension in interactivity brings the touch screen interface back on par with the level of interactivity afforded by a mouse, and allows for more expres-

sive, and more importantly, more traditional user interface designs to be practical.

To achieve this solution I propose to augment the touch screen with a commercial software application [13] and 1 or 2 web cameras (can be relatively low cost) to allow the visual identification of hands, and their position relative to the touch screen. This visual detection layer allows for the use of standard touch screen applications and hardware and, when compatible hardware is detected, can enable the additional functionality using a simple software switch.

The use of stereo vision allows for an approximation of depth (or a 3D co-ordinate to be determined when using proper stereopsis), enabling the use of simple depth based gestures such as zooming by moving ones hands in and out from the screen. This can be done by looking at the direction of movement on each camera. If both cameras see the hand moving left, then the hand was moving left. If the right camera sees left movement, and the left camera sees right movement, then the hand was moving out from the screen, and if it is opposite, the hand is moving inwards. When opposing movement is detected, the result can be averaged to keep the correct location of the hand, and the difference can be used to determine the depth. Face detection and tracking was also proposed initially, but was later abandoned due to the complexity of implementation. It is also worth noting that the proposed system can be used to augment non-touch screens in a similar manner.

This project was carried out in conjunction with Unlimited Realities, and will be developed for their Umajin^{TM} engine, to allow the rapid development of software that makes use of my solution for end users. This will also make the various image processing techniques and algorithms, as well as the near touch detection

available for any who licence the use of Umajin^{TM} or Fingertapps^{TM}.

The Umajin^{TM} engine is the base product that Unlimited Realities have created to serve as the engine that drives their main line of products. It is provides a wide range of functionality such as 2D and 3D processing and generating of images, as well as displaying them and allowing the generating of Microsoft Windows interfaces and applications. Umajin^{TM} also provides a scripting language interpreter that allows for the development of applications that use the Umajin^{TM} engine. These scripts are called U scripts.

Built on top of the Umajin^{TM} engine is Fingertapps^{TM}. Fingertapps^{TM} is a framework developed by Unlimited Realities specifically for the development of touch screen applications. Fingertapps^{TM} allows programs to handle touch events, and includes a collection of predefined constructs for interface building (called widgets) such as buttons, sliders, and Unlimited Realities application wheel[14]

1.2.2 Use Cases

Single camera

With a single camera (as shown in 1.1) the system will be capable of detecting the users hands and enabling non-depth based gestures. In this setup the system will suffer from reduced accuracy as the user moves further from the camera as well as error introduced by deviation from the defined plane depth.

Dual camera

The optimal setup for the system is as shown in 1.2. The user will be able to use the full range of implemented gestures, and have accuracy from both sides of the

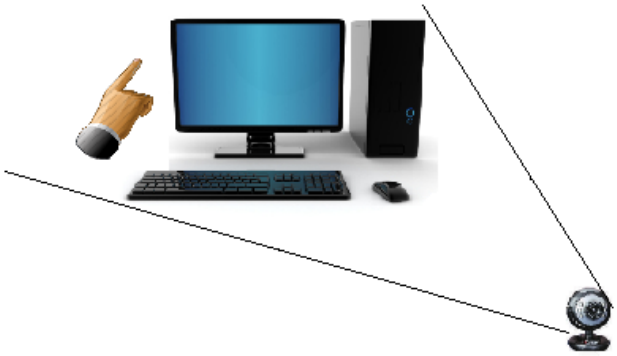


Figure 1.1: Single camera setup for system

screen, as well as depth input.

Non-touch enabled screen

Although the motivation of the project is to overcome the inherent limitations in interactions with touch screen systems, my proposed solution can also be used to provide an alternative or extra input system for traditional computer systems. The user will experience all the benefits of gestural inputs (the range of which depends on if the system has 1 or 2 cameras present), but will still require a mouse/keyboard for normal input. This limitation may be overcome using similar methods to typical touch based systems such as fully gesture based interfaces (depth perception can be used to emulate a touch) and screen based keyboards for textual input.

1.2.3 Advantages of the Proposed System

Touch based interfaces are quickly becoming a popular method of human-computer interaction due to their high level of intuitiveness and tactile feedback during use. Devices such as the iPhone^{TM} and iPod^{TM} Touch are quickly becoming the de

Figure 1.2: Layout of proposed system

facto standard in the consumer electronics industry, as well as other commercial areas such as retail units and advertising/customer service kiosks. Current touch interfaces however have the constraint of only allowing interaction when the user is making physical contact with the screen, which severely limits how the user can interact with a system (the typical design is to make the touch event analogous to a click with a mouse, and to use actions such as double clicks and click and drags as the primary actions). This is highly restrictive for both the user and the interface designers, and can limit the practicality of the system.

The proposed system has the advantage over other touch based systems of allowing interface design that closely mirrors the design of traditional computing systems. This means there are fewer limitations on how the user may interact with the system, and designs may follow strongly established conventions for user interface design and allow new and unfamiliar users to instantly pick up the new system due to its familiar interface design that is highly analogous to both the traditional mouse driven interface and the contemporary touch interface. The

proposed system also has the advantage of having a unique design in a market that is quickly becoming saturated in touch or image recognition based systems of comparable designs.

Another advantage of the proposed system is that it should be reasonably robust from the user perspective with regards to camera placement. Because the intended implementation does not use true stereopsis it does not require the cameras to be in a known location to calculate accurate 3D coordinates. While most information of the cameras placement can be calculated from the image detected, information such as scale is not so easily determined. While accuracy of the proposed solution is somewhat dependant on the cameras being symmetrically placed and centre aligned with the screen, it is hoped that the solution will still prove sufficiently robust from the user perspective without the need for carefully measured and placed cameras.

1.2.4 Disadvantages of the Proposed System

Due to the nature of touch screen interfaces certain concessions must be made, such as the need for large object designs, and also the inclusion of an object snap or 'lock-on' system with some form of hysteresis when nearing objects to compensate for the input jitter likely to be experienced within the proposed system. Other than these concessions the general methods of user interaction with the system may still stay be traditionally designed. This makes user interface design much easier as there are no longer major hurdles to overcome due to the lack of any ability to discern the users area of interest without a 'click' (an action generally reserved for selections and important actions, and that can be highly disruptive to the flow of a user interface).

Because the system will not be using true stereopsis it will not generate true 3D position data. Although this was done to provide a simple solution that has increased robustness to non-measured camera placement and minimise the need for user set-up and calibration, it has the draw back of reduced depth information. While a general measure of scale and direction is gained for depth, further information is required to convert that data into an accurate point in 3D space. This limits the usefulness of the depth data to a degree and means that when used in a User Interface context, visual feedback becomes very important for the user to gauge what they are actually doing within the system.

The system also has the slight draw back of requiring the user's hands to be held up near the screen. After prolonged periods of use this may lead to fatigue of the user, although it should not be significant when compared to normal touch screen use as that also requires the user to reach out and touch the screen. It is also worth noting that although physical touch is not always a required interaction within the proposed system it is still required that the users hand be near the screen or suffer reduced resolution. As such the user cannot use the system from more than a few feet away without employing expensive and high end digital cameras with a very high resolution, potentially limiting its range and usability by those without access to said equipment. There are also inherent drawbacks in the requirement for the user to be able to point with their finger for the physically impaired who may be unable to complete such a gesture. It is hoped though that the detection will be robust enough that, with the aid of a well designed UI with large interface objects, that the physically impaired might still be able to use the solution to a degree.

1.2.5 Goals of the Project

The final goal of this project is to have integrated a set of image processing functions into the Unlimited Realities Umajin^{TM} engine that may be applied to images of any source within the engine, and to use these functions to create a near touch system that can be successfully demonstrated using a Graphical User Interface. The key milestones and features that would denote success are;

Demonstrate the Ability to track the users hand in front of the screen

To achieve this goal the system must be able to clearly separate the users hands from all other background information in the presented video input(s), and do so with a high degree of reliability for the user.

Display Ability to report the point they are gesturing too with sufficient accuracy for general use

The system must be able to report the location that the user is pointing to with a reasonable level of accuracy. This level of accuracy must be sufficient for the user to navigate a menu system and highlight objects of interest. This can be made easier through intelligent user interface design.

Recognise simple gestures

The system should be able to recognise simple gestures such as linear motions of the users hands with minimal input delay, allowing the user a more varied control method.

Display the implemented system in action using a graphical user interface

All of the features planned should be presented on a graphical user interface in a usable system. The user interface should be designed in such a way that it

can effectively demonstrate all the systems features to a user, and highlight the advantages offered by the systems design over typical touch screen systems.

1.2.6 Contributions of the Project

The key contribution of this project is the successful marrying of the concept of touch based UI systems and the flexibility and freedom of traditional UI designs made with the traditional mouse and keyboard in mind. This is an important step in UI development as touch screen devices have quickly become the de facto standard in the consumer hardware industry with smart phones, hand held gaming consoles, tablet's and even consumer desktop PC's to a lesser extent all having adopted the system. The lack of a non-touch interaction is extremely limiting on both their usefulness and to User Interface designers who must carefully design UI elements around the inherent limitations of the system. By adding a non-touch input level to the system it is hoped that interactions with the system will no longer be bound to the event of a touch, allowing designers to create more flowing menu systems, tool tips that allow the user to investigate the consequences of an action before committing to it and more.

Further to this improvement of UI flexibility with touch devices it is also hoped that the system will provide a simple system that is robust enough to be successfully used in a home consumer environment without the need to careful placement and calibration of expensive 1st party hardware. The user should be able to simply place one or two cheap web cameras in approximately the right position as space and practicality allow and have the system work with as little need for calibration and set-up as possible.

Chapter 2

Existing Related Work

Many existing systems have been designed to attempt to overcome the inflexibility of touch based systems, ranging from the tethering to a touch sensitive physical layer, to the problem of sufficient efficiency of inputting typical commands such as text entry and menu navigation. Some of these solutions are explored and evaluated in this section.

2.1 Microsoft Surface

Figure 2.1: Microsoft Surface in use, showing device recognition.

The Microsoft Surface^{TM} [7] was designed primarily to be an interactive re-imagining of touch screen systems. It has a much broader focus on integration with other devices such as cameras and smart phones, and was designed to be a very interactive experience for the user. The system itself is set up as a 30 inch touch screen under an acrylic top, designed to be used like a table. It has 5 near infra-red cameras under the acrylic top to detect objects that reflect the near IR illumination shone at the surface. It uses this system, coupled with image processing to track touch objects, and to recognise patterns that mark and track objects. The original concepts showed users placing devices such as cameras or smart phones on the table surface and having the driving software detect these objects and interface with them wirelessly in a context sensitive manner.

For example, a customised setting for use in a bar was demonstrated by Microsoft on the Surface^{TM} where users were able to view menus, order drinks and view media from their table surface. When the drinks were delivered the surface was able to keep track of their drinks and display information about the drink (useful in a wine catalogue or something similar) as well as individually labelling and tracking every drink so that it would not get mixed up with other peoples drinks. In another example, when the camera was placed on the surface and a ring appeared around it, indicating the device had been recognised and was now connected to the surface. This ring tracked the movement of the camera on the table and allowed the user to open up a menu and transfer images from the camera to the surface wirelessly.

These tailored menus allow for a large amount of custom interfaces and device specific usability, however they possess the drawback of only being available on supported devices. This can be a serious drawback for the product as it requires

a large amount of third party support in order to be successful. Due to this it is highly likely support for certain devices, such as the popular apple iPhone^{TM}, may be dropped in favour of pushing Microsoft's own products (in the given case of the iPhone^{TM}, the Microsoft Windows Phone 7^{TM} would be pushed instead), leading to vendor lock in and anti-competitive behaviour[15].

While the Microsoft Surface^{TM} can be a powerful tool in specific situations such as restaurants, bars and kiosk stands, it's dependency on third party customisation and support means it's use in general circumstances is limited to that of a typical touch screen system, albeit tipped on its back, and larger than the usual touch screen.

2.1.1 Summary

Advantages

- Works by interfacing with physical objects and communicating wirelessly with it.
- Uses customised, context sensitive menus for everything.

Disadvantages

- Requires high degree of third party support for device capability.
- Still has limitation of touch only for interface design.
- Only practical for kiosks and tailored situations.
- Impractical for general computing due to its large physical size.

Figure 2.2: Prototype of Project Natal.

2.2 Microsoft Kinect (Project Natal)

At the 2009 Electronics Entertainment Expo (e^3) Microsoft™ announced a new input device/game controller was in development for their Xbox 360™ console [2]. This project was given the code name 'Natal'. The intention of this new hardware device was to provide an experience to rival the innovative Nintendo™ Wii[1] console and its use of a controller designed to replicate the form of the common remote control. It contains an accelerometer, allowing motion to be tracked in 3D space with reasonable fidelity (motion along an x,y,z plane set was very accurate in the original version, but the Wiimote lacked the ability to accurately detect rotation, a problem that was rectified with the release of the Motion Plus attachment for the Wiimote [16]). Because of the large amounts of critical acclaim and financial success the Wii was experiencing due to its unique motion control, Microsoft™ decided to implement their own version of motion control to attempt to claim a portion of the market share.

Microsoft's solution takes advantage of a highly capable and flexible combination of an RGB camera and a 3D range finding camera[17] to create a point

cloud of any object within its field of view at ranges from 0.8 m to 3.5 m. At a range of 2m it has a resolution in the x/y directions of 3 cm per pixel, and in the z direction of 1 cm per level and possesses a field of view of 58° in the vertical direction, and 40° in the horizontal direction. This 3D camera is mainly what makes the Kinect so unique, it can create detailed 3D models of whatever IR reflective objects are in the scene it is presented with and track their motion in real time. This is achieved by projecting a structured IRlight pattern onto the screen, and using the IR CMOS sensor to view the distortion of the structured light pattern and use that to determine the depth for that pixel. This can be further added to the RGB camera output to give a textured 3 dimensional view of the scene, as well as an approximate skeletal model. Combined with body tracking techniques and facial recognition coded into the driver/software development kit, it allows for accurate tracking of the user in 3D space and when combined with appropriate software, can provide for interactive and 'true to life' interactions. There is also facial recognition software built into the Kinect to allow for individual users to be remembered and have customised interactions/expression recognition.

All of this was demonstrated by Peter Molyneux in action during his Milo demonstration[18], where the user interacts with a virtual boy who is able to recognise the user visually, determine their mood (using audio to detect the tone of voice used), carry out conversations and even interact directly with the user. During one demonstration, the user drew a picture for Milo and 'gave' it to him by holding the picture out for him at the top of the screen in front of he camera, allowing Milo to 'take' the picture from the user, which became digitally rendered as an object within the demo world.

While this level of interactivity and accuracy is noteworthy, for the application of

general user interaction in a workspace environment it would have a major drawback. While this kind of interactivity is ideal for games and entertainment where the user typically would use the system for less than 1 to 2 hours, in a working environment where the user would typically be using their computer constantly for several hours at a time, it becomes much less practical as the level of physical exertion needed to interact is much larger. Although this is an interface design choice, and the hardware could be suitable for use in the solution proposed in section 1.2, the field of vision (FOV) of the hardware may be insufficient to be used appropriately as an augmentation for touch screens without placing the detection unit a reasonable distance from the screen and off to the side. This can cause problems with objects on the desk causing occlusion, as well as placing desk size requirements on the system. The minimum operation distance of $0.8\,m$ further increases this problem. It is also an issue that the hardware is a recent development, and therefore is quite expensive in comparison to simple web cameras, which are a well established technology [19].

The z-camera used in Kinect is able to produce depth images at a resolution of 640×480 pixels, at an average latency of $40\,ms$ ($25\,fps$) and take full RGB images at resolutions up to 1600×1200, although no frame rate is given for video streams at any resolutions in the technical specifications, the manufacturer, Prime Sense^{TM} (early prototypes used a 3DV z-camera, but it failed to meet Microsoft's requirements), provide these for the identical PrimeSensor^{TM}.

2.2.1 Summary

Advantages

- Accurate z-camera.
- Allows for detailed detection of objects, motion and more in 3D space.

- Allows for facial recognition etc.

Disadvantages

- Requires physical exertion, more room to move and a large screen etc.
- Currently very new technology, and very expensive compared to web cameras.

2.3 Sony Play Station Eye

Figure 2.3: Sony PlayStation Eye.

The PlayStation™ Eye Toy™ (known as the PlayStation Eye™ in its latest version for the PlayStation™ 3) [20] first debuted in 2002 at the Electronics Entertainment Expo as a technological demonstration and was subsequently released in 2003. It was developed as a type of virtual/augmented reality interface device to take advantage of the PlayStation™ 2's (at the time) substantial processing power [21] and a range of image processing techniques to detect the users body.

Using full body recognition and tracking meant that the PlayStation™ Eye Toy™ was capable of allowing a wide and diverse amount of different gestures and it came with a large amount of different mini games in the form of bundled software called 'Eye Toy: Play'. In the following year Play 2 and 3 were released to further bolster the Eye Toy compatible games line up. These mini games were a set of activities ranging from boxing and Kung-Fu fighting, where the user threw punches and kicks at a virtual opponent who could dodge them and launch counter attacks, to a fast paced table tennis game where the user challenges a series of increasingly difficult players, through to a series of 'catch' type games where a group of objects is set loose moving all over the screen and the user must gather up as many of these objects as possible.

Due to the recent influx of competing motion controller's (devices that use the users physical motions as the games input, as opposed to button presses on a controller) from Sony's competitors in the console market (the Nintendo™ Wii with its new motion plus Wiimote add-on and Microsoft's new Kinect™ system) Sony developed its own system using a wand like object topped with a white ball that can be internally illuminated multiple colours, and visual detection using the PSEye to compete in this area. Their solution has a high degree of accuracy and allows for a fully 3D mapping of the controllers location, including rotation (something the Wiimote only gained through the motion plus add-on). Sony calls their solution the Move™.

Outside of its gaming oriented use, the latest version of the PlayStation™ Eye has seen some use in academic circles due to its High quality, High speed web camera in a relatively low cost package [22]. Its ability to process resolutions such

as 640×480 at high frame rates(greater than 60 frames per second), coupled with its possession of a microphone make it a powerful and versatile USB camera that allows for high speed image processing at a much lower cost than typical high speed camera configurations found in these systems.

Being a web camera type device, it has some inherent drawbacks with respect to the defined problem. The first is one of the more obvious problems; the current use of the PSEye of the system requires the user to be standing and interacting in front of the camera. This does not solve the problem, it is simply moving the limitations in interaction from one plane to another. It also introduces an extra element of physical exertion that makes the system impractical for extended use periods (most likely why the games that accompany the PSEye are short casual games that can be played for as short a time as the user requires, as opposed to the longer, more involved games that typically garner the best review from critics/journalists).

2.3.1 Summary

Advantages

- Uses image processing to isolate and interpret point's of interest as controls.
- Allows for user input at a distance from the screen.

Disadvantages

- Doesn't always solve the problem touch interaction inflexibility, usually just moves it into a new plane.
- Some gesture based systems allow for both cursor and click functionality .

- Has the drawback of requiring a large area for user movement.
- Requires a large screen to make everything visible and easy to interact with from a distance.
- Requires extended physical effort, making them less practical in environments requiring long periods of use.

2.4 Don't Click it!

Figure 2.4: Don't Click It! website.

The Don't Click It website [23] is the product of German Communication and Design student Alex Frank of Essen-Duisburg University. The intent of the project was to create a new user interface design that challenged the traditional concepts of user interface design, the 'click' in particular. He set out to create an interface for a web page that allowed the user to navigate the entire site without the need for ever clicking a single button (apart from the entry button). This was achieved through the use of roll over menu expansion and containing all information in the website as though it was a sub item in a menu, or by grouping items as info

graphics.

Gesture input is also featured on this website, in the form of a small case study in the Explore→ Button Lab menu tree. This exploration is limited to two simple gestures to activate a button, one that sweeps the width of the button, and one that circles it. The first is described as inadequate as the user can easily activate the button in this manner unintentionally during navigation. The second is described as inadequate as the user must be shown how to perform the gesture before they may use it due to the unintuitive nature of the gesture. A timed hover is also proposed as a method of implementing click-less interactions, but is described as being unsuitable as it slows down user interaction as they must wait for the timer to activate the button they desire. It should be noted that all these methods attempt to recreate the functionality of a click, without actually using the mouse button. By using a click in this case they would achieve all of the functionality they are trying to achieve, with none of the stated drawbacks. Indeed the only real drawback allowing a click would provide in this context is that it violates the concept of the website.

A useful feature of this site is that it tracks various user statistics such as user satisfaction with the interface design and the frequency with which users click when the user does not require it. At the time of the last visit, the site had recorded a total of 3,444,574 visits since its inception. of these visits, in random polling 1,325,002 users claimed they did not miss the click convention of traditional user interfaces, and 754,018 users claimed they did miss the click. These figures give a roughly 64% approval rating among users. From these users a total of 2,009,234 unnecessary clicks were detected during browsing the website. Of these clicks, 817,724 were recorded as being on purpose (the user intended

to make the click, having forgot it was not required), and 761,847 clicks were accidental (the user subconsciously clicked the button without thinking). These statistics show almost half of the clicks recorded and commented on were a result of the deeply ingrained convention of the click in interface design. It shows that clicking has become an automatic response to objects like buttons or menu items.

It is worth noting that there were several flaws with the information gathering. The responses relied on user honesty for accuracy, and the user was not always polled on matters such as their approval of the site, or the reason for their errant clicks. Also the wording of the question on if clicks were accidental or on purpose may be unclear to users, as any click could be considered an accident in an interface that does not make use of clicks.

The number of positive responses gained from users shows that there is potential for the increased use of click-less browsing in user interface design, however the number of negative responses also shows that an interface based completely on hover events and no-click interactions is not an ideal system either.

2.4.1 Summary

Advantages

- Showcases a new style of user interface for desktop computers based entirely on cursor position, no clicking required at all.
- User opinion shows that this philosophy of user interface design is accepted by the users.
- Supports gestural inputs to replace the click function in some circumstances.

Disadvantages

- Gestural input is used to replicate a click, but a click would solve all these problems the proposed solution introduces with no drawbacks other than violating the concept of a click free interface.
- Gestures are very simple, and can easily be activated, possibly with the user not even being aware they have done so.
- A 36% disapproval rating from users is far from ideal.
- Careful user interface design is needed to make up for every intersection of an object and the cursor being an interaction.
- Frequent re-arranging of the user interface makes navigation difficult in many instances.

2.5 Mid-Air Input Systems for Projected Displays

Projected displays are now the de facto standard for a wide range of tasks such as meetings and seminars as they allow the dissemination of ideas easily. An evolution of the projective displays that is now being explored is the ability to interact with them. This makes them a more effective communication medium, and in the presence of multiple input devices, can make them ideal for collaborative efforts. They follow on from the legacy of blackboards and whiteboards, but the ability to interact with them has been the last major barrier that has prevented them from completely replacing them in all aspects of life.

The simple pointing device is currently the most common input device for projected displays, and is often found in the form of a laser pointer. This allows the user to project a visible dot onto the screen, allowing them to highlight, visually,

certain areas of the screen. This is of limited use on its own as it does not allow the user to alter the contents of the display, only to act as an indicator for viewers. Multitudes of different solutions have been offered for this problem, but as of yet, none have been formally adopted by the general end user.

Some of the solutions explored include a glove device covered in sensors detects the movements of the user in 3D space, and allows them to replicate the motions of a cursor and mouse[24], using visual recognition to detect touches on a physical glass screen that acts as the projection surface[25], and in the area of text entry, several methods were explored by Shoemaker et al[26].

2.5.1 Textual Input

In the research of Shoemaker et al, 3 proposed text entry systems were tested for use with very large wall displays. First was a character wheel that displayed all 26 letters in a circle formation, and allowed the user to select one by guiding and angle indicator using a pointing device to line up with a character, and then have it selected with a button press on the pointing device. Second was a QWERTY keyboard interface that displayed a virtual keyboard on the display, and allowed the letters to be selected by pointing device and button press. Finally a 3x3x3 cube with all the letters of the alphabet was proposed with the user selecting one of the 9 individual elements on each of the 3 visible faces (giving 27 choices). The element was selected with a pointing device and button press.

Of these proposed methods, the circle and QWERTY input schemes were found to be the most practical text input systems, with the QWERTY proving the

fastest and most accurate at shorter distances. At a distance of 18 feet however, the circle input scheme proved to be superior to the QWERTY system, with input speed becoming comparable at the greater distance, and error rate increasing much faster with the QWERTY system. At short distances both gave similar speed results to a stylus based touch QWERTY keyboard, or handwriting based inputs (approx. 20 words per minute).

While results were good for a large wall display, such text input methods would be much less practical in a touch enabled system designed for typical home use. Due to problems such as the size of text on screens, and the size of screens them selves, it is often impractical to be located more than arms reach from the screen in home systems. As user distance increases, concessions must be made such as trading screen real-estate for increased font size. Because of this close proximity, the fastest and most practical text input system is still a keyboard. A physical keyboard allows for the average user to input text at speeds in excess of 30 words per minute[27], and does not require the use of display space for the text input as a touch screen soft-keyboard or the proposed mid-air input systems would.

Summary

Advantages

- Does not require a keyboard for text input.
- No keyboard means greater freedom as there is no wire to tether the user and no unwieldy input device to carry around.

Disadvantages

- Much slower than using a keyboard due to greater need for movement, which will likely be especially notable for skilled typists.

- still requires some kind of input device for pointing, which will lead to significant fatigue when creating large documents.

2.5.2 Single/Multiple Point Cursor Input

There have been several methods of cursory input explored in the field of augmenting or replacing traditional touch screen technologies. Such efforts involve work by various researchers on the use of infra red point detection to determine cursor location for input. A common implementation of this in modern research is the use of the Wiimote controller for the Nintendo^{TM} Wii[28, 29, 26]. This is used frequently due to its low cost, common availability as a commercial off the shelf product, and high versatility, with its ability to sense infra red, measure movement along it's x,y and z axis using an accelerometer.

There are two common configurations of the Wiimote. The first is to use the Wiimote's IR camera to detect any reflected IR light and use this to determine the input. To make this practical the user must be located sufficiently far from the IR source that only the IR reflective object (whether it is the users hands, or some kind of 'wand' like pointing device) used for input is detected, but not so far that not enough light is reflected to give reliable input. To improve the range of this the input object can be augmented with a reflective material like mirrored glass or reflective tape etc. or the IR source near the sensors position can be removed, and instead an IR emitting input object can be used instead.

The second common configuration for input with the Wiimote is to use the 3-axis accelerometer to track movement in 3D space. A drawback of using the accelerometer to sense movement is does not properly track motion along all the degrees of freedom of the controller. The Wiimote is able to detect motion in

either direction along it's 3 axes, but it is unable to differentiate this motion with rotation, and the altered direction of gravity's action after rotation. This short coming was compensated for with the addition of the motion plus accessory, however this is not a part of the standard Wiimote, and thus requires an additional purchase to be made, and the accessory to be attached to the Wiimote in order to gain its benefits.

When using the accelerometers the user can get an accurate flow of data representing motion of the controller, however this must be tracked in software to determine how this data can represent a real world position relative to the display as only motion is tracked, and not any real relative position by the Wiimote accelerometers. When configured using IR light and the Wiimotes IR sensor multiple objects can be tracked in software to act as inputs in the position of these objects can be interpreted in what ever way the system designer desires to gain meaningful inputs from them.

Summary

Advantages

- Allows for pointing actions to be made with and object similar to a remote control, which is instantly familiar to any user.
- Using accelerometers means that motions are not limited to the field of view of a camera or IR sensor as with a pointing device.
- Is fairly distance independent, allowing operation at a wide variety of ranges.
- Requires little image processing for point detection as the IR sensor essentially receives a brightness map that can be thresholded.

Disadvantages

- Requires an extra device for pointing when used as a pointer, or some kind of IR reflective marker on the hands to give a reasonable range.
- Limited battery life.

2.6 Camera Based Input Detection

Another method of input detection common among the various projective display based systems is to use a camera to detect inputs from the user. This can range from simple controlled point detection on a surface[30] to full image processing of scenes for augmented reality systems[31],[25]. Both of the solutions have merits, but also drawbacks to be considered.

2.6.1 Augmented Reality

Augmented reality is a exciting area of research in where 'reality' as the user perceives it is augmented by computer generated content. This can range from simple things like information tags and pop ups, to virtual entities that can be interacted with. This augmentation, especially when coupled with the information resources of the internet, allows for every aspect of life to be enriched for all people simply by wearing some kind of augmented reality interface such as a set of glasses with a built in camera and displays to overlay the computer generated content over what the user sees. A draw back of augmented reality is that it is limited to a first person view, or a second person representation (the most common of which is to have the camera in front of the user so the user perceives the augmented reality as if they were looking in a mirror). While this is fine for single users, it makes the solution far less practical for multiple users in environments such as collaborative work environments.

The first person perspective implementation of augmented reality is a very enticing area of research due to the implications it could have on every day life. With integration into the internet as well as possibly local storage, it would be possible for users to do various activities such as record moments of their life from their own perspective for later reference, get information on anything they look at in the form of a floating pop-up information bubble, or to interact with virtual objects. All of this can be achieved using visual processing to recognise real-world objects, and activate appropriate context sensitive actions on recognition, or rendering 3D virtual objects onto the desired location within the users view by determining where it should be placed in 3D space. This is usually accomplished by recognising a specific marker within a scene and determining its position and orientation, and applying this translation onto the 3D model of the object to be rendered.

The second person implementation of Augmented reality can be seen in examples such as the work of Kreuger et. al., Videoplace[31], Although this system can also be implemented as a virtual reality interface as well, where the user is overlaid in a generated scene, as opposed to generated content being overlaid on a real scene. Videoplace was originally developed to act as a video feed back communications application where users could interact with each other, but later evolved to cover many applications such as the CRITTER interaction, where a small virtual creature engaged in basic interactions with the user in various ways. It would chase, or be chased, climb up the user and jump around in various situations, dependant on environmental factors such as the users stance, or the user could shake off the CRITTER.

Input techniques for augmented and virtual reality are as diverse as they are numerous, and there are a great many input methods that can be considered.

Summary

Advantages

- Can be used anywhere, at any time.
- Can be customised easily for individual users.
- Gives ready access to information.
- Uses many image processing techniques useful for solving the problem of touch interaction restrictions.

Disadvantages

- Enriches user interaction with reality via augmentation, but does not facilitate any interaction with computer devices.

2.6.2 Image Processing Based Input Detection

The image processing solution to input detection is now becoming increasingly common with the prevalence of low-cost yet high quality digital image capture devices as with only two of these cameras full 3D tracking of an object can be accomplished to sub-pixel accuracy in some cases using stereoscopy, as seen in[25]. Alternative uses of image processing for input detection have also been made such as Andrew Wilson's Touchlight system [30].

Touchlight was a projective display on a semi transparent material that allowed the display to be projected in front of the user while they were illuminated by an IR light source. Two IR pass filtered cameras were placed behind this semi

transparent plane to observe the user, and their inputs were perspective corrected with respect to the semi-transparent surface and the results averaged to give a brightness map of the user for input. Because the users hands were significantly brighter when they were touching the surface, thresholding allowed inputs to be gained from this brightness map, giving full multi-touch and any appropriate surface with only the need for 2 cameras, a projector and an IR light source.

The Bell Labs Gesture VR system developed by Segen and Kumar[25] used a dual camera system with the ability to detect the users index finger and thumb in 3D space and the angle the user is holding them at. This information was then used to either act as a cursor input, or used to implement simple gestures, depending on the context of the application.

Several applications were developed to showcase this input system; A 3D scene builder that allowed the users to build scenes by drawing out primitive shapes and then manipulating their position in 3D space to create more complex scenes. Navigation in this application was gesture based, or alternatively could be menu driven. There was a landscape fly-by application what allowed users to aerially navigate Yosemite Valley using their position of the hand relative to the centre of the table in a forwards/back direction control velocity, angle between thumb and index control the angle of turn (with the centre at the middle of the thumbs travel, so turns could be made in both directions) and the tilt of the hand control the pitch. Finally, A single camera with gestures was used to act as a controller for the video game Doom, where there was a gesture for the ‘shoot’ action, and movement was generated based on the position of the hand relative to the centre of the table the camera was set above.

Summary

Advantages

- Can be very cost effective with modern digital cameras being low cost and commonly available.
- Can be highly accurate in appropriate situations.
- Gives full 3D interaction.

Disadvantages

- Processing costs can be more intensive than other solutions.
- May require control over environmental conditions for reasonable operation.

2.7 Feature Comparison

The table 2.5 shows a rough comparison of the features presented by the explored existing solutions to user interface design, particularly in relation to their usefulness within the proposed problem space. As can be seen, the Microsoft^{TM} surface is the only solution to rely on touch user input, and this restricts it in it's applicability. The other techniques which are not reliant on touch are useful in some situations but not others. The home use suitability column represents the current suitability of a system based on it's cost, development and ease of use/implementation in the context of general computer interaction. This will be subject to change with techniques like Augmented reality and the Microsoft Surface becoming much more practical as their costs decrease and the become feasible to combine with other techniques.

Solution	*Input Device Required*	*Touch Interaction*	*Non-touch Interaction*	*Cost Effective*	*Suitable for General Home use*	*3D spatial awareness*	*Requires Physical Exertion*
IP input detection	no	no	yes	yes	yes	yes	yes
Augmented Reality	no	no	No	no	no	yes	no
Single/Multi cursor input	no	no	yes	yes	yes	yes	yes
text input	no	no	No	yes	yes	no	yes
Clickless interface	no	no	yes	yes	yes	no	no
PS Eye	no (yes for 3d)	no	yes	yes	no	yes (with move controller)	yes
Kinect	no	no	yes	yes	no	yes	yes
Surface	no	yes	No	no	no	no	no

Figure 2.5: Comparison of Features and capabilities of various current solutions.

Chapter 3

Design and Implementation of Proposed Solution

3.1 Design

Throughout the design phase of the project much consideration was put into possible use cases for the system and how it would be best to solve the problem of limited degrees of freedom. It was also considered that the product must be suitable for use by typical consumers and must be marketable in a commercial environment. Originally the concept was to create an image processing based system that detected the user and provided them with a gesture based, touch free interface, as seen in the movie Minority Report. The Minority Report style interface has been seen as a benchmark for human computer interaction since the movie was released, and many researchers and corporate research and development labs have been pursuing the creation of a system like it. Due to the maturity of touch screen technology at the time they were seen as an ideal starting point for these designs, and so products like the Microsoft Surface^{TM} and Hewlett Packard Touchsmart^{TM} began to emerge. Many research oriented facilities such as universities who did not possess a requirement to make profit chose to focus on the creation of more ambitious and cutting edge systems based on

visual feedback from the user such as the work by Johnny Chung Lee[32] with his head and multi-finger tracking systems built around the infra red detector in the Nintendo Wiimote^{TM}. Some of the devised solutions did not even require a dedicated screen, and could be used anywhere via a small projector.

After some initial research was conducted, it was decided that many systems such as the image processing based 'touch free' systems were already in development (eg. [33, 34, 35, 36, 37]) and thus there was no need to add to the growing list. It was instead decided that, given the large amount of commercial proliferation of touch screens in the market place, a solution would be developed to bridge the two different solutions, allowing existing touch screens to be used, while allowing an extension system to be layered on top of them, giving a touch free interaction layer on top of the existing touch interface. Incorporating this touch free layer on top of the touch interface allows an extra degree of freedom to the user, in which they are no longer tethered to only having a single action available to them (to touch or not to touch, that is the question). This freedom not only benefits the user by giving them a much richer experience, but also is a tremendous benefit to user interface designers as it gives them much more freedom in the way they design their interface, allowing them to once again make use of conventions such as hovering menu expansions and tool tip pop-ups, conventions that had to be abandoned due to the inherent nature of the touch screen.

3.1.1 System Design

The basic design proposed in section 1.3 consisted of a system that was capable of detecting the users hand through a camera system, and from this information correlating the hands position to a position on a screen. This would allow the user to use their hand as both a cursor input device by pointing to a desired

location and as a gestural input device by recognising certain patterns of cursor movements as a pre-assigned command. To implement this system require 3 major steps; First to have a method of determining the users hand position, then to have a method of converting this to a cursor position relative to the screen, and finally, to have a gesture recognition engine of some variety to distinguish gestures from normal cursor movement.

Hand Detection

There are numerous methods of detecting a users hands and their orientation that vary by accuracy, robustness and efficiency. Because one of the intentions of the system was to work for as wide a range of consumers as possible in real time, methods that work based on ideas such as shape detection or require a large amount of computation were unsuitable. Shape detection would be a limiting factor to the potential customer base as it puts the requirement that the user posses no serious deformities such as missing fingers or an inability to make certain shapes with their hand. Another point to consider is that looking specifically for objects that can be identified as hands can preclude the use of other limbs as input devices. For these reasons a skin detection system was chosen as the method of detecting user input.

Skin colour based detection is a robust and reliable method that could be used to detect human input for this system across a range of conditions. Few objects in the world share similar colouration to human skin, and the few that do can easily be singled out to users and avoided in the set up of the system (for example, wood tends to share similar colour to human skin and so users could be

instructed to set up the system in an environment where wood would not be in the cameras field of vision). Human skin also has the useful property of being easily segmentable from other objects in certain colour spaces such as Y'CbCr and HSV/HSL (there is debate over which is the more appropriate version of the colour sapce. Using this method over methods like Corner detection coupled with pattern recognition meant that the final solution would be much easier to implement and would have much better performance on older, less powerful machines. It also means that users who are physically impaired and not able to use their hands for input may still use other limbs which may not be, such as pointing with their feet instead. One major drawback of using skin detection over other solutions is that it precludes the user from obscuring skin on the input hand. Items such as gloves must be removed to the system, and in the case of the physically impaired using feet, shoes/socks.

Converting Input to Cursor Data

Two possible setups were conceived of for the system, a single camera variant, and a dual camera variant. With a single camera the user would have the ability to use their hand as a cursor, and could use 2 dimensional gestures. A single camera would have the draw back of a lack of depth sensitivity leading to errors as the user deviates from the designated plane of operation, as well as errors of greater magnitude as the user moved further from the side the camera was set up on. This error is due to the fact that the screen position must be marked by the user on a certain plane relative to the screen. If the user strays inwards or outwards from this plane on the single camera setup, the camera will perceive this as a movement left or right (depending on which side the camera is viewing from). To compensate for errors caused by deviating from the designated plane, a second camera can be added to the system.

This second camera would be placed on the opposite side of the screen, and thus, the error induced movement would appear opposite to the other camera (as long as both cameras are placed symmetrically). The average detected position of the two cameras would therefore represent the real position in the plane of the hand, and the difference and its polarity of the two detected positions would indicate the depth and direction from the plane respectively.

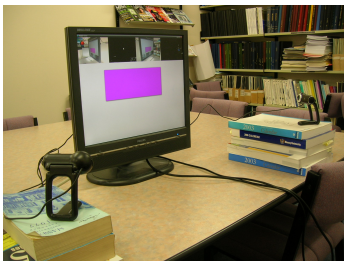


Figure 3.1: The system running in dual camera configuration.

Camera positioning is also important in the proposed system for three reasons. The first is that if the cameras are placed too close to one another, then they will be closer to the middle, and thus the users hand will provide significant occlusion of where they are pointing, making accurate detection harder as the cameras move closer to the centre. The second reason is that the cameras must be placed in such a manner that as much of the screen is displayed as possible. The more area the screen occupies in the cameras field of vision, the more resolution can be gained for the user input. If the camera is at a very steep angle to the screen then resolution will become poor, especially farther from the camera and the system will become much less reliable, to the point of being unusable in extreme cases. Finally, the cameras should be as symmetrical as possible, as the less symmetrical the setup of the cameras, the greater the difference each camera will perceive when deviating from the plane of use. This will reduce accuracy by

introducing a left/right bias when deviating from the plane.

Using the detected position of the users hand as input, the first step of the system was to determine the borders of the screen within the cameras field of vision. This information should only be needed once as the screen and cameras position should remain fixed during use. This can be achieved by simply having the user point to each corner of the screen in a set order and designate that location by a button press. This allows the user to specify their preferred near touch depth and lets the system know where its area of operation should be within.

The next step once the screen location is set, and position of the users hand had been reliably detected is to convert this location reported by the camera(s) into a pixel location on the screen output (this should be done before depth from plane is determined). By doing this, the user's hand movements can be converted into a cursor position, facilitating the near touch interaction of the user, and the use of gestures.

Gesture Implementation

With the ability to for the user to input a cursor position implemented, the next step is to create a suitable gesture recognition engine. The idea of using the mouse cursor as a method for inputting shape based gestures has been present for a while, being featured in the internet browser Opera since 2003[1], the game Black and White (released in 2001)[2], along with other places. The gestures work by moving the mouse in a pre-defined motion such as a spiral or a cross to activate the related action. This functionality can be emulated with touch screens

[1]Opera Web Browser Press Release on Mouse Gesture Inclusion, Last Visited on 24/07/09, http://www.opera.com/press/releases/2003/04/11/

[2]Black and White website, last visited 22/07/09, http://www.lionhead.com/bw/Default.aspx

by following a touch and matching it to a pattern. This however can be awkward for the user due to the friction of their finger on a touch screen, the imprecise nature of touches and the possible need for complex direction changes in the gesture leaving the user feeling self-conscious of their exaggerated actions and how external viewers may perceive them. These problems can be overcome by the proposed system using certain design concepts and a bit of thought in interface design.

Because the system does not require direct touches from the user for input (touch can be emulated using depth sensitivity), friction and the need for the user to be less than arms length from the screen is removed from the system. To address the need for the user to use the system with a minimum of fatigue and to not feel self conscious waving their arm about to initiate gestures, a simple set of contextual gestures were decided to be the most suitable approach. To facilitate quick implementation the gesture set was designed to require a minimum of computation to detect. The model for the gestures was to be a simple linear motion within a region of interest. This model was tested within the soccer demonstration for suitability as described in section 4.2.3. Designing the user interface appropriately means that the regions of interest can be set easily in contextually relevant positions, and the simplicity of the gestures makes it simple to average out noise while maintaining reliability of recognition.

The gesture set decided upon was a forwards/back gesture, a scroll up/down gesture, a zoom in/out gesture and a close gesture. These were selected as there are common actions taken in many typical activities such as web browsing, reading documents and exploring folders. The gestures are explained below;

Forwards/Back

The forwards/back gesture is a very common action taken in navigating within web browsers or in presentations, among other things. It is an action analogous to turning a page in books, and it beneficial to implement a gesture in such a way that the user will intuitively link the action to the gesture, providing a more natural and intuitive response from the systems user interface. Following from existing design conventions, the forwards/back actions should be activated from the top of the window, where the corresponding buttons are located in existing user interfaces. To achieve this a region of interest can be set up at the top of the window that will trigger a forwards/back motion whenever a linear directional movement is made within in the appropriate direction, right to left (forwards) or left to right (back). To ensure general navigation does not create false actions, gestures can be required to be over a certain length, and to not deviate in the vertical plane by more than a set amount.

Scroll Up/Down

Similar to the forwards/back gesture, scrolling up and down is a common event present in applications that allow the user to browse documents, and is analogous to moving a page up or down to move the section the user desires to read into their field of vision. The convention for scroll buttons is to be present on the right side of the window, arranged in a vertical manner running the length of the window, or document being viewed. Setting up the region of interest on the hard right of the window has the advantage that the user will have nothing to interact with on the far side of it, and so will never cross over it (unless the application is running in a non-full screen manner, which it is not currently designed for). This means that less care is needed for the mitigation of false activations, as the user should not be entering this region unless specifically intending to scroll.

The actual gesture itself can be implemented as a simple sweep of the hand up or down, to 'drag' the document in the desired direction. Scrolling down could be achieved by the user sweeping their hand from bottom to top, and scrolling up would be top to bottom. This gesture could be tied to a scroll bar control along the right of the screen, which not only gives the user visual feedback on their scrolling, but follows established convention as well.

Zoom In/Out

The zoom gesture presented an opportunity to showcase the systems depth sensitivity by allowing the user to zoom in or out by moving their hand closer to or further away from the screen. By moving their hand towards the desired point, the user would zoom in as if they had moved closer to this point. By the user moving their hand away from the screen, they could zoom out as if they had moved further from the screen. This gesture could be preformed on any point on the screen not already reserved for another gesture, to allow the user to zoom on a specific spot, or if noise made it too difficult to use the system without accidentally zooming, then it could be limited to a region on the centre that required the user to make it obvious that they were intending to make a gesture as suggested in [38].

Close

The close gesture could be implemented by having the user make an obvious cross slash gesture across the whole screen, from any corner to the opposite corner, and the mirroring slash. The action of 'crossing out' an application is easily related to removing or rejecting something. Because this is a large gesture that requires the user to traverse the entire screen in one go, meeting each of the four corners, it is unlikely the user would accidentally make such a gesture during general use, and thus it is acceptable to use the entire screen as an activation region for the gesture.

Finally, a user interface would have to be designed to demonstrate the system adequately. To properly indicate how the system could be used, as well as how it would be advantageous to traditional touch screen systems, the user interface was to be created from a traditional touch screen application, and modified appropriately to incorporate the features the new system allowed.

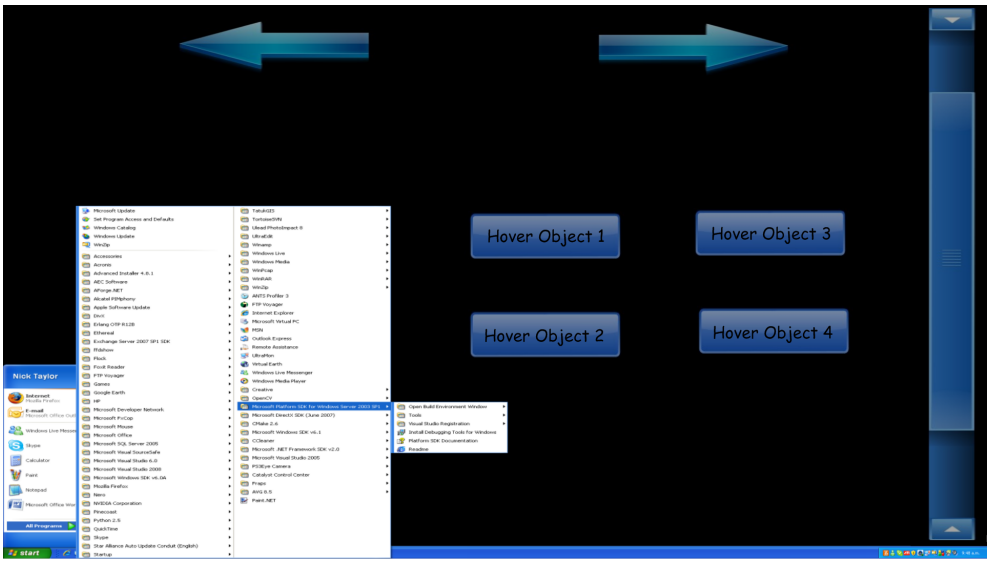


Figure 3.2: Mock up of possible user interface showing a menu that may be navigated, objects that reveal tool tips when hovered over and highlighted gesture regions. These are designed to highlight the benefits of the system.

3.2 Implementation

Once the design intentions had been finalised the method of implementing the design specifications had to be decided upon. During development it was requested by Unlimited realities that a playable game be made to act as a technical demonstration of the implemented image processing algorithms and video streaming systems within the Umajin^{TM} Engine. This demonstration was to be shown at

the CeBIT consumer trade show. For this reason it was decided to create a soccer game that would utilise camera feedback in the same manner as the Near Touch User Interface system that would serve both as an engaging and entertaining technology advertisement for consumers visiting Unlimited Realities CeBIT booth and as a demonstration of the feasibility of the intended gesture engine for the Near Touch system.

3.2.1 Soccer Technical Demonstration

The final form of the technology demonstration had two core sets of requirements that it had to achieve. The first requirement set was that it was engaging and entertaining to potential customers of Unlimited Realities and that it showed potential uses of the image processing and video feedback systems integrated into the Umajin^{TM} Engine during the course of this research. The second set of requirements (and the ones that are pertinent to this thesis) were that the demonstration should adequately show that meaningful information could be obtained from tracking a single input point and that the information could be interpreted as a meaningful action within the context of the system. To best show this the system should be as close as possible in it's underlying mechanics as the actual Near Touch User Interface system created. To this end a soccer game was decided upon. Soccer is one of the most widely played games in the world and is instantly recognisable and enjoyable to most people, fulfilling the need for the demonstration to act as good advertising for Unlimited Realities. The act of kicking a ball is a similar gesture to many of the proposed simple motions for the Near Touch Interface, and from this gesture the information of how hard the ball was kicked (from the velocity of the foot motion) and the vertical angle of the kick can be determined and used as input to simulate the kicking of a virtual ball within the game. By implementing user input in a similar manner to the

Near Touch Interface (a single point acting as a moving cursor) this can fulfil the second requirement set and demonstrate the feasibility of the proposed gesture system.

The system design was similar to the Near Touch system in that it had the same 3 main steps; Detect the user input (in this case the motion of their foot), convert this information to a meaningful position within the system and to recognise this information as a gesture with meaning and information associated to it.

With a camera observing a designated region for the gesture to take place image processing could be used to determine the location of the foot and segment it from the background, making tracking it a simple task. Assuming that there are no foreign objects moving in the region reserved for the demo players (reasonable given the fact games generally occur in a controlled environment to avoid disruptions or external influence) then background subtraction is a simple method for removing the background from an image so that only the motion of the kick gesture remains for consideration. The fact that the "imaginary ball" the user will be kicking to generate the gesture will reside on the ground and the camera will be observing this region further reduces the potential of moving objects as only a narrow field of vision is needed and the camera can be aimed down at the ground so other surroundings are not visible.

There are many forms of background subtraction[39] that can be used for motion detection ranging from simple frame differencing (anything that did not change between frames is considered background) to statistical modelling[40] of a video stream to work out what is typically present at each pixel. All the various methods have their benefits and drawbacks, frame differencing is very quick and will

pick up any differences between two frames, but this is also a drawback as motion will leave 'holes' behind as the object moves as show in in 3.3.

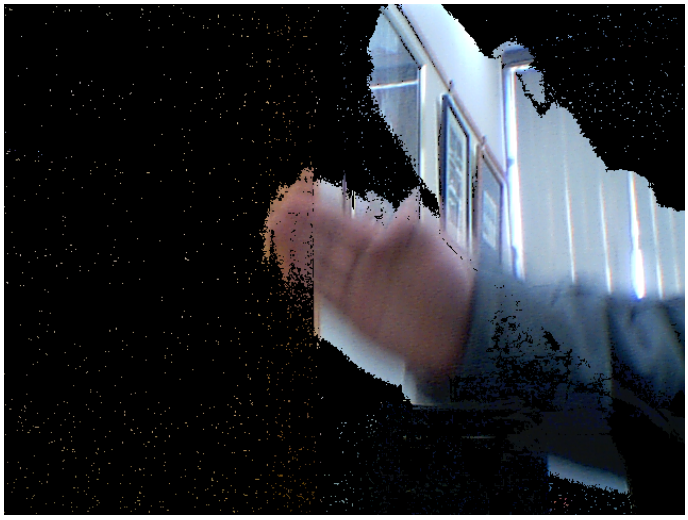


Figure 3.3: 'Hole' left behind by a waving arm when using frame differencing.

To solve the problem of holes some kind of model can be used to 'learn' the background of a scene. For the purpose of this tech demo it was decided that a weighted Gaussian average of previous frames would be used to determine the background, with more recent frames having a higher weighting. This has the benefit of requiring no more memory than frame differencing while offering the additional ability to remember backgrounds. The strength of this memory can be adjusted via the weighting, creating stronger resistance to holes. The cost of a better memory with a running average is that the background subtraction requires a warm-up period to develop an accurate view of the background.

With the background removed the next task is to determine the motion of the kick. The remaining portion of each frame after background subtraction consists

Figure 3.4: using weighted averaging to 'learn' the background leaves no holes behind moving objects.

of the players leg and foot, but the foot is the only relevant part of the image. Because the placement of the camera can easily be controlled, and the region the algorithm observes can be easily set it was decided the best way to only view the foot was to observe only the region the foot would enter. This meant that only a small portion of the leg would be included in the Binary Object (BLOB) representing the detected motion, and thus would not have much effect on the detected position. Some conditioning is also applied to the image in the form of dilation and erosion of the image to fill holes and remove small amounts of noise. As a simple method of converting this BLOB to a trackable point on screen it was decided to use the approximate centre of mass. There are many methods of approximating the centre of mass for basic shapes such as using the most pixel dense row and column as the x,y co-ordinates for the point, for this system the centre of a bounding box of the BLOB was chosen as it seemed to give the better noise resistance than row/column density during testing.

For the final step, by tracking this information while it is either still within the region of interest or still moving forward and then looking at the average velocity and average angle of the kick after the point of the 'ball' parameters for the kicking action within the game engine can be calculated (allowing for actions like scaling of input to give a better feel for the game). For the game a logarithmic scaling factor was applied to the kick power as it gave a good sensitivity range for the reasonable powered kicks required to score a goal but compressed the extreme kicks so that they all cleared the stands. Without the scaling the kick power was difficult to tune without there being obvious discrepancies between the effort exerted and how far the ball went on screen or a large portion of the power range being feeling useless as a slightly hard kick could clear the stands and an extreme power kick could send the ball sailing into what appeared to be outer space.

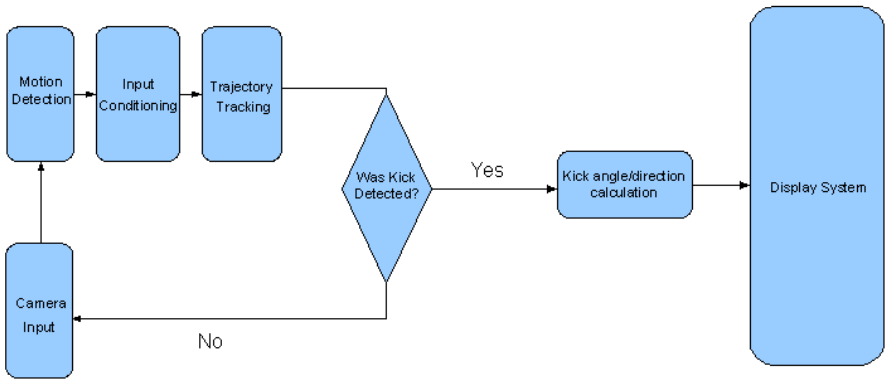


Figure 3.5: Flow diagram on the soccer demonstrations operation.

3.2.2 Near Touch User Interface

The final version of the system to be implemented was the actual near touch interface layer. This consisted of several parts; the Visual detection of the hand in 3d space via camera, the calculation of the detected hands position relative to the users screen and finally, the representation of this information in a usable graphical user interface.

For the detection of the users hand it was decided that the most suitable method of detection would be to search for skin in a manner similar to [41]. Skin detection is a reasonably simple, yet robust system for detecting the presence of a human. It is usable over a wide range of skin tones and lighting conditions, however it requires that skin is not covered up with clothing in order to be detected successfully (so for this system the user cannot be wearing gloves). Lighting effects can also be reduced by including lighting correction. One way to implement this correction is to convert the image into a colour space that keeps the luminance and chrominance channels separate such as HSL or HSV and normalising the luminance channel. Doing this will remove lighting variations such as shadows, leaving only consistent colour behind. One drawback of this method is artifacts such as diffuse colour bleeding may be left behind where shadows were.

The only other issue with using skin detection to detect user input is that any skin coloured objects in the background may lead to false positive detection. This could possibly be compensated for using methods such as background subtraction on the region outside of the screen. It is also necessary to subject the skin masked binary image to erosion and dilation to connect slightly broken regions that should be connected and fill holes. Further conditioning was done by culling all detected distinct objects that were not of a specific size to help reduce noise.

(a) Uncorrected image of an orange.

(b) Lighting Corrected, notice the orange diffuse colour bleeding where the shadow was.

Figure 3.6: Lighting correction applied to an orange.

It is reasonable to assume that any detected skin is the hands as the cameras will be focused on the users screen and, optimally, that will occupy the vast majority of the screen real estate, so the only thing that should be passing the cameras field of vision is the user's hands as they interact.

Once the user's hand has been segregated as an identified object the next step is to detect its position in the scene as a pixel co-ordinate. This location is approximated by taking the upper left or right corner of a bounding box of the hand object with respect to the camera the footage came from (left camera uses the upper-left corner, right uses the upper right). This makes it reasonably accurate and quick to compute an approximation to the point the user is pointing too provided the cameras are placed below the centre line of the screen and at a sufficiently acute angle to the screen (angles must also not be too steep or the accuracy and resolution of location detection at the far edges will suffer). There is a draw back from using this method in that the user must hold their hand at a certain orientation relevant to the camera position. If the users hand is too flat and the camera is low then the top of their hand may be detected as the top of

the bounding box, leading to the cursor displaying higher than the user expects, this is something that users can easily adapt to. There is also a positive in this as because the system does not specifically look for a human hand in a pointing gesture, it allows users with physical limitations that could preclude them from pointing with a finger to also use the system, as it is still functional with relatively vague inputs if the user is able to adjust to the offset of the output cursor.

From this point we can now determine the borders of the screen and gain the required parameters to run perspective correction, allowing us to more accurately translate the detected position of the users hand into a location on the screen. The basic principle of perspective correction is that a point (x,y) may be transformed to the corresponding point on another plane (x',y') using the equations:

$$x' = a_{11}.x + a_{12}.y + a_{13} \div a_{31}.x + a_{32}.y + a_{33} \tag{3.1}$$

$$y' = a_{21}.x + a_{22}.y + a_{23} \div a_{31}.x + a_{32}.y + a_{33} \tag{3.2}$$

If a sufficient points are known then a set of simultaneous equations can be derived and solved. This means that if you do not know the values of the co-efficients a_{11} to a_{32} for the transformation but a set of four points between the two planes is known (or can be inferred from other information) then the co-efficients for the transformation can be calculated. To calculate the variable a_{33} an extra pair of co-ordinates is needed, however for this system it is not required to know the exact scale transformation as all points are converted to pixel co-ordinates on screen, so only the ratio of pixels in the relevant portion of the image to pixels on the screen need to be known. In this case a value of one for the scaling co-efficient a_{33} can be assumed to be 1. To get these four points the user will run through an initialisation process for the system that involves pointing to each of the corners of the screen and indicating them with a key press. From this we gain four known

points in the source image so all that is left is to attain the corresponding points in the transformed image.

Because the system will calculate the cursor position based on relative measurements in the image space as opposed to real world positions we can assume the positions of the corresponding points based on certain criteria. One point can be assumed to be stationary, acting as a reference for the other points. The best point for this is the one closest to an upper corner of the image, this will be the upper right for the left camera and upper left for the right camera due to the rectangular shape of the screen. From this point the lower point for that side can have it's co-ordinates set at the same x-position as the reference point and it's y-position at the same as the other bottom corner (to give the screen the greatest amount of presence in the image, improving resolution). This leaves the y-positions of the far two points able to be assumed as the same values due the fact computer screens are rectangular and have parallel sides. Finally, the x-coordinates for the two far points can be calculated by taking the difference between the top and bottom y-positions and multiplying them by the aspect ratio (which in turn can be derived from the screen resolution) and applying a this as an offset in the appropriate direction. The main weakness of this method is that it relies on the resolution being at the same aspect ratio as the screen's physical borders. In practice this should not be an issue however as mismatching resolution and screen aspect ratios leaves the image stretched undesirably and with LCD screens, non-native resolutions result in the image being blurred substantially as the size of pixels is no longer a 1:1 ratio between image and physical display.

One limitation of this method of transforming image space co-ordinates to screen

space co-ordinates is that it will only work within the plane specified by the user when pointing to the four corners of the screen. As the user deviates in and out from this plane calculated values will become inaccurate. Because cameras are placed at an angle to the screen it will also introduce error as the user moves to the further side of the screen from that camera due to the lower resolution of physical screen to image pixels. While the problem off plane deviation cannot be easily dealt with, adding the second camera to the system will assist with the lowering resolution as the user increases distance from the camera. Because the camera is placed at the opposite end, as one camera looses accuracy the other will gain it as the user comes closer. This effectively averages out the error introduced by distance. The other benefit of having a second camera is that depth information can be inferred from the difference between camera images. As the user deviates from the use plane they specified each camera will perceive the opposite motion. The left camera will see the users hand move right when they move towards the screen and the right camera sees a left movement. Using this a magnitude and direction can be taken from the difference and used as approximate measure of depth change for basic gestures. The gesture system will be as described in the previous section, tracking the linear motions of the input within specified regions of observation.

3.3 Results and Accomplishments

This section will describe the resulting system and give both quantitative and qualitative measurements of the Near Touch User interface system implemented.

3.3.1 Soccer Technical Demonstration

The design for the soccer technical demonstration was to have either one or two cameras focusing on a point of interest to represent the virtual location of a soc-

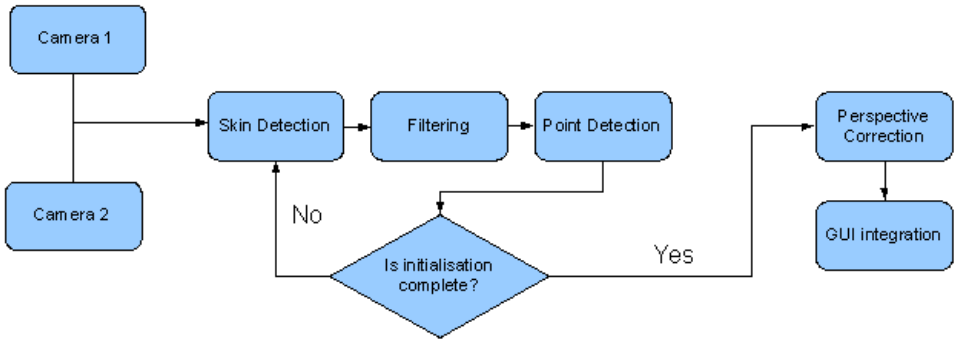


Figure 3.7: Flow diagram of proposed system. This will run once per frame.

Figure 3.8: Final soccer demonstration complete with animated goal keeper and direction indicator

cer ball. The user would then make a kicking gesture and the motion of that gesture would translate into the user kicking the virtual soccer ball in the desired direction. Ideally the user would be able to control direction, height and power of the kick. The initial concept was to have one side on camera for detecting the height angle of the kick, and a second camera looking from approximately 45 degrees above (to avoid occlusion if the user leans over the ball while kicking). Early in development it was decided to turn the demonstration into more of a game, by introducing a goalie into the design, making the accuracy of the kicking

system more obvious and introducing a higher level of interactiveness to the system. The goalie was given full 3D movement and a set of animations for saving the ball or attempting to save it (as well as a fail animation if the user missed the goal completely, or failed to even reach the goal). Due to this decision (as well as time constraints and the fact that video input was still in the process of being developed in Umajin™ when the demonstration started being prototyped) it was also decided to remove the second camera for sideways angle detection and to simply introduce a timing element to the technological demonstration by having a moving arrow at the bottom of the screen rotating from left to right constantly.

As discussed in the implementation section, for detection of the kick gesture, all motion was segregated and then each distinct object detected was labelled, conditioned using erosion and dilation, and finally objects were culled based on size (if an object was too small then it was discarded as noise). A bounding box was placed around the region of interest where the virtual ball was located, and as soon as motion entered the bounding box it was recorded until it left the box again, or stopped for a certain period of time. Motion can be detected in both directions allowing for play from both directions, and angles are made absolute, so that. The direction of the kick was input at initialisation so that the user could kick from any side. The movement of the foot was determined by summing the number of pixels in each row and column and finding the most dense of each and taking that to be the centre of mass (the centre of a bounding box could optionally be used instead, although this was found to be a worse approximation through subjective visual comparison).

The detection of kicks, and generating a numerical output for angle and velocity was all prototyped within OpenCV, and once completed was ported to Umajin™

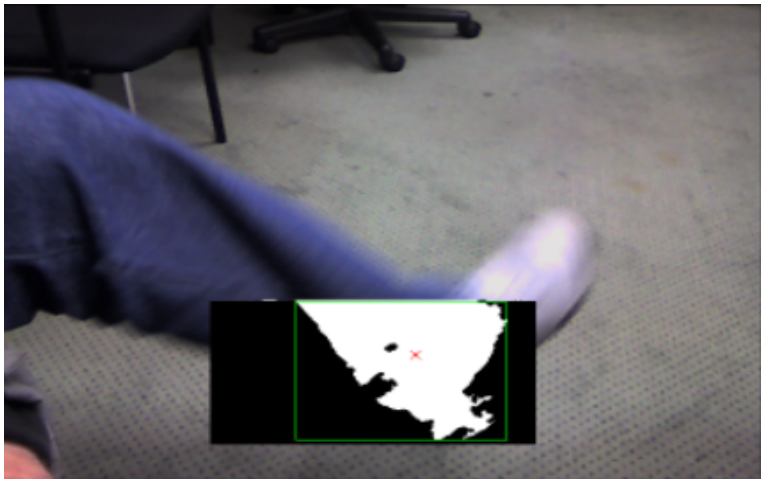


Figure 3.9: System Detection of a kicking motion within the area of interest.

for integration with the game and animation code and resources. This was because at the time work was begun video input had not yet been completely implemented within Umajin^{TM}and OpenCV proved the most suitable prototyping environment. OpenCV[42] is an open source computer vision library created by Intel to give researchers a full set of computer vision tools and algorithms to allow them to quickly build complex solutions to computer vision problems. It also served the purpose of demonstrating the computational advantages of Intel's proprietary instruction set, Integrated Performance Primitives[43]. The primary reason that OpenCV was chosen over hardware accelerated solutions like nVidia's CUDA[44] or OpenCL[45] are that both of these solutions are hardware dependant, and at the time of implementation the hardware was not wise spread, being less than 2 years old and still a high-end product, CUDA especially is a proprietary nVidia API and will only run on certified nVidia hardware (G8X and above based GeForce and Quadro GPU's or Tesla computers). OpenCV also had the benefit of being set out in a similar manner to Umajin^{TM}, images are laid out as one dimensional arrays and both are primarily written in C++ (although

many wrappers are available for OpenCV including Python[3], Java[4] and even FORTRAN[5]), improving the portability of all code written.

Once prototyping had proven successful running off a pre-recorded web-camera video in OpenCV and video stream input was implemented fully in Umajin™, the final system with 3D video feedback of all input was implemented. The final implemented system is shown running in 3.8, although it should be noted that this is a debug build and that the camera output with processing would not be visible in the actual demonstration.

The kicking detection worked well and was fairly intuitive, allowing for both soft kicks that barely rolled forward at all, to large kicks that sailed off into the distance and out of the stadium (power was scaled logarithmically in the end as it allowed for easier control by expanding the low-mid power range, and compressing the high power range). It also allowed enough control that, with a bit of practice the user could chip the ball over the goalies head if he came far enough forward. In terms of performance, the average run-time for the soccer demonstration was approximately 5 ms when there was no activity, and increased to around 15-20 ms when the user kicked. This allowed a frame processing rate of up to 50 fps to be maintained, well within acceptable speeds. This also allows the input to benefit for higher frame rate cameras, as they will capture the movement more smoothly, and give better average values due to the non-linearity of organic movement.

[3]Python Wrapper for CUDA home page, Last Visited 24/07/09, http://mathema.tician.de/software/pycuda

[4]Java libraries for CUDA page, last Visited on 24/07/09, http://www.gass-ltd.co.il/en/products/jcuda

[5]FORTRAN compiler page for CUDA, last Visited on 24/07/09, http://www.pgroup.com/resources/accel.htm

The solution proved to be both a suitable demonstration of the suitability of simple gestures for controlling simple inputs using basic detection, and a popular showpiece at the CeBit 2009 trade show where it was shown by the Unlimited Realities marketing team[46]. The demonstration also successfully proved that a simple gesture recognition system based on linear motions within a specific region could be implemented using video for user input.

3.3.2 Near Touch User Interface

Figure 3.10: Developed system being used. Debug information is shown at the top of the screen, this would not be present for the typical end user.

The system currently is functional though a very simplified GUI for testing that allows for the user to move around a cursor, select a large object in the centre of the screen, and a small button in the lower right corner. There is also functionality to allow the testing of both positive and negative depth changes from the specified plane, although no specific functionality has been assigned to these

actions yet other than changing the colour of the centre object. It is able to be set up with either a single camera, or two for depth functionality and improved accuracy. The user may then specify the calibration plane for which the system will be used, and the system is then fully set up and ready to be used.

Calibration can be at any depth level from the screen, or even on the angle if the user does not sit directly to the front of the screen being used. The optimum position of the plane is in line with the screen at approximately 1 inch from the screen frame and parallel to the camera plane. This allows the best resolution from the detection algorithm while still giving room to make use of the depth change testing accurately. If depth control is not required, greater accuracy may be achieved by moving the plane of calibration closer to the screen. Depth perception as it is currently implemented in the system is also currently quite reliant on the calibration plane being approximately parallel to the plan the cameras are placed on. As the plane is rotated at an angle to the camera plane a noticeable bias left/right is introduced to the depth reading. While this bias is not sufficiently large to become an issue until the use plane deviates noticeably from the camera plane this does cause restrictions to use conditions where the system can operate optimally. These conditions can usually be mitigated easily, but it is still preferable to have the system operate perfectly under all conditions.

Once the system has been calibrated the user is presented with the user interface and a cursor representing the point of the screen they are pointing too. Currently the User Interface consists of a box in the centre of the screen that changes colour based on depth information (if the user deviates sufficiently in or out of the defined use plane) or if the user rolls the cursor over the box. The purpose of this object is two fold; First it is to simulate a larger selectable item

as would typically be designed in a touch based UI. Secondly the box acts as an indicator to replicate feedback for the 'click' gesture based on the users depth position. The other major element on the screen is a smaller button that changes colour on rollover. This button is to simulate finer object selection from the user, showing the feasibility of selecting smaller UI elements with the system and demonstrating accuracy.

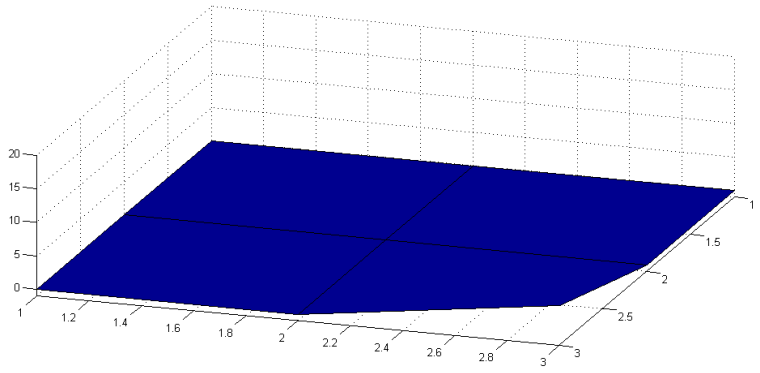
Accuracy Analysis

To determine the objective accuracy of the system a series of tests were conducted at various plane depths from the screen, starting from flush with the screen (0 mm) and moving outwards in 10 mm increments up to 50mm. Error was measured from these planes in both the Y and X directions (vertical and horizontal) at nine points on the screen; the 4 corners of the screen, the 4 midpoints of the screen edges and the centre of the screen. An approximate error range was also taken when deviating inwards from the designated plane of use at plane depths of 30-50mm and for all depths in an outwards direction. For these tests the left camera was placed 520mm left of the screen (from it's centre) and 370mm forward of it. The right camera was placed 990mm right of the screen and 270mm forward. The reason for the large distance difference between left and right cameras is because they were different models, with the right camera having a much lower field of view, requiring that it be placed further away to encompass the entire screen. Both cameras were operated at a resolution of 640x480 capturing a 19" Phillips LCD monitor with a resolution of 1280x1024 and physical dimensions of approximately 380mm by 300mm.

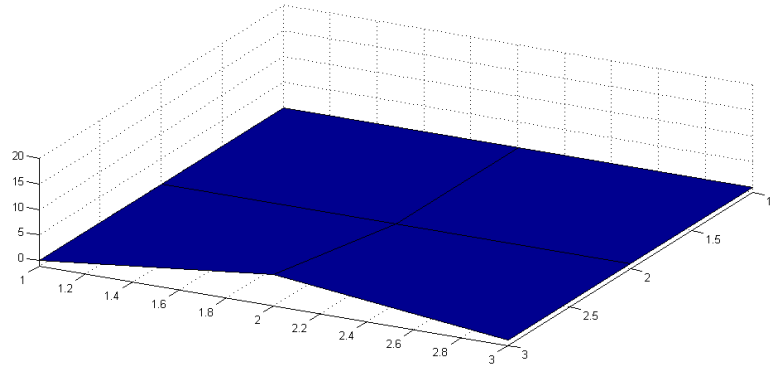
All figures showing the measured error have the magnitude of the error on the vertical axis as estimated in 5mm steps (i.e. 0mm means no appreciable error,

5mm means 0-5mm of error, 10mm means, 5-10mm of error etc.) with the point (1,1) being the upper left corner of the screen and (3,3) being the lower right most corner. It should also be noted that obtaining accurate measurements at small magnitudes is difficult, as they can easily be drowned out by small amounts of temporal noise from the environmental lighting and camera sensors etc. Error levels are also highly dependant on the accuracy of the user during the calibration phase. While all efforts were made to ensure the most reliable calibration, human inconsistency will always introduce some error into the system.

When the screen plane is set as the frame of the screen the error for the system should be at it's minimum as the correlation of detected position to screen position is the closest. Because the frame acts as a physical reference for the user it is the most simple to keep in line with and also gives the most robust calibration as touching the physical corners gives no user error when calibrating. As can be seen from these results in 3.11 there was little appreciable error, with only the lower middle and left corner giving a noticeable amount of error. With a deviation out from the plane of 10mm no discernible error was detected in the x direction. In the y direction there was 10mm of error introduced. This is primarily attributable to the cameras being positioned at level with the bottom of the screen, causing a straight outwards movement to also appear to move upwards from the cameras perspective. This is a weakness of the current system where camera placement can cause directional bias of error, particularly in the vertical axis. This error can be minimised by having the cameras placed symmetrically and at approximately the same height as the mid-level of the screen (although the bigger the screen, the more error will be introduced on the vertical axis). When the plane error is expanded to +20mm the observed cursor error was between 5 and 15mm in the x direction, depending on cursor position, and



(a) X error

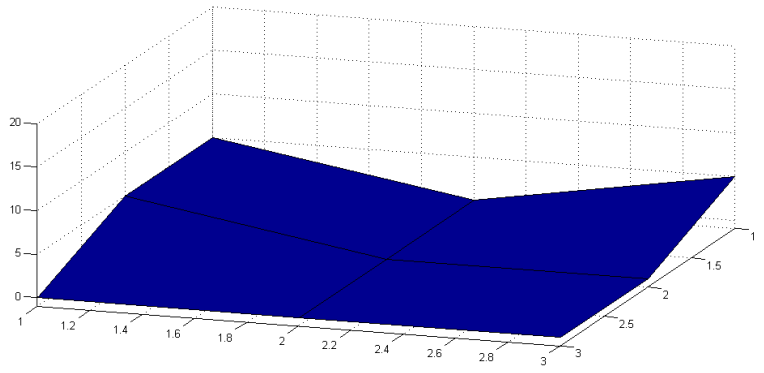


(b) Y error

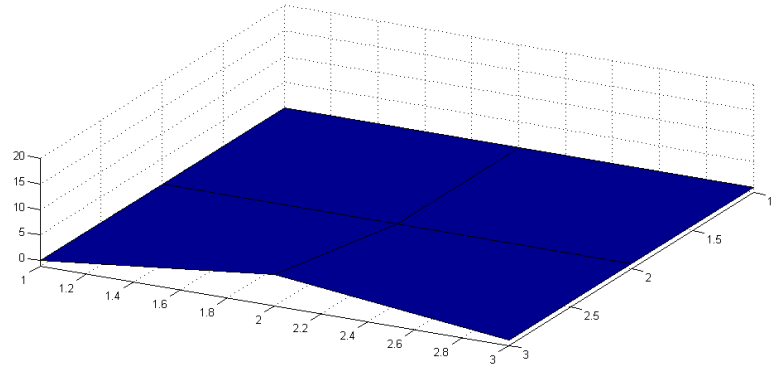
Figure 3.11: Error with plane set at screen frame.

the y error was observed at approximately 15mm. With a deviation of +30mm from the calibrated plane the error becomes quite significant with a horizontal error ranging from 20-30mm and a vertical error of 10-40mm. With this large deviation from the plane vertical error becomes significant enough to hamper use due to it's high variance across the screen. Horizontal error, while notable, is still relatively easily compensated for by the user.

Having a plane depth of 10mm from the screen frame gave similar results as



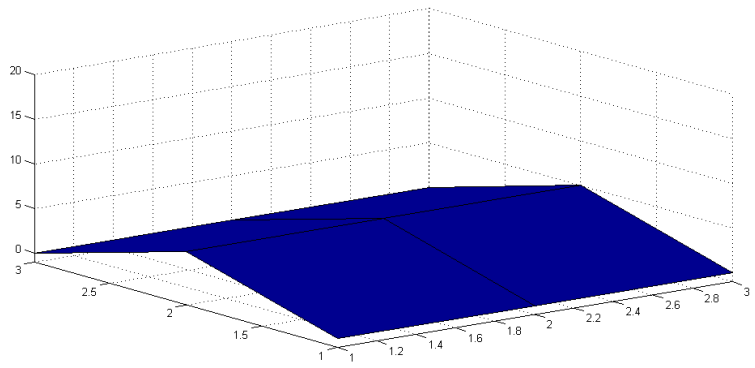
(a) X error



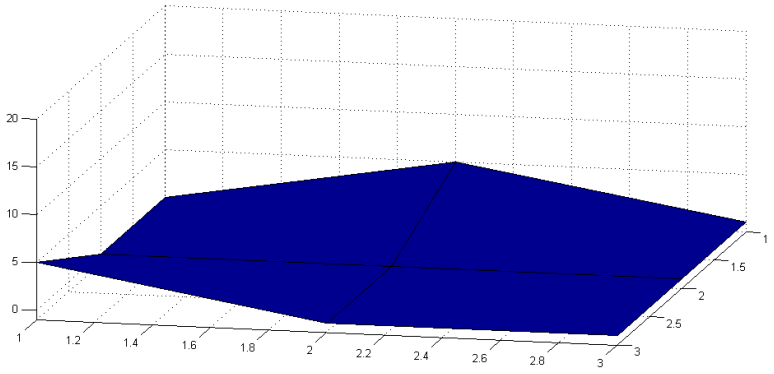
(b) Y error

Figure 3.12: Error with plane set at 10 mm from screen frame.

shown by 3.12, with only the extremities from cameras (upper left and right corners) having any notable error, along with some small error in the y direction around the centre of the lower edge. Similar results were observed as with a plane set to the screens frame when deviating from a 10mm plane distance. Results were actually slightly better at +10mm and +20mm than the previous plane distance, likely due to inconsistencies when calibrating. At +30mm there was slightly more appreciable error with a horizontal error of 30-40mm, although the vertical error was slightly smaller at 10-30mm.



(a) X error

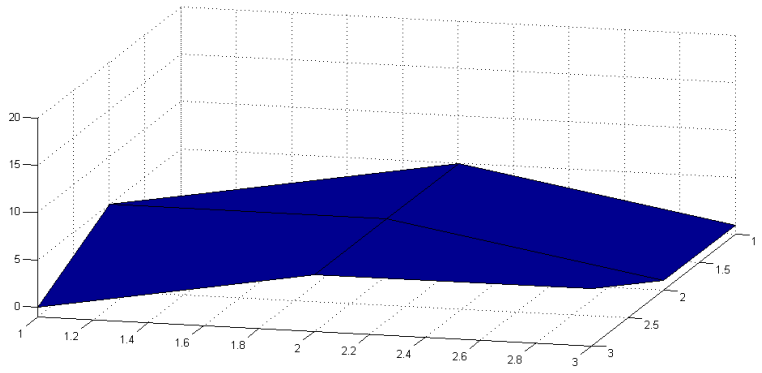


(b) Y error

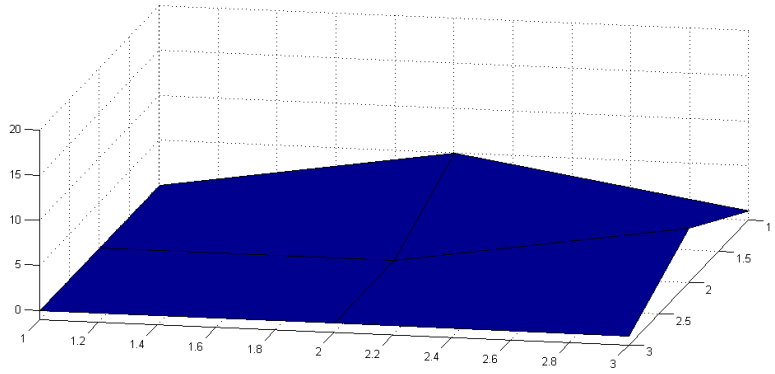
Figure 3.13: Error with plane set at 20 mm from screen frame.

At 20mm (figure 3.13) there is some more notable inconsistency, although it is still fairly minor in magnitude. In the current system it is still difficult to notice over small temporal jitters cause by noise from the cameras. Similar errors when deviating from the plane were noted as with the previous plane calibration.

A similar situation can be seen in 3.14, some small error is noticeable in the x direction along the middle of the screen and lower right corner, but not enough



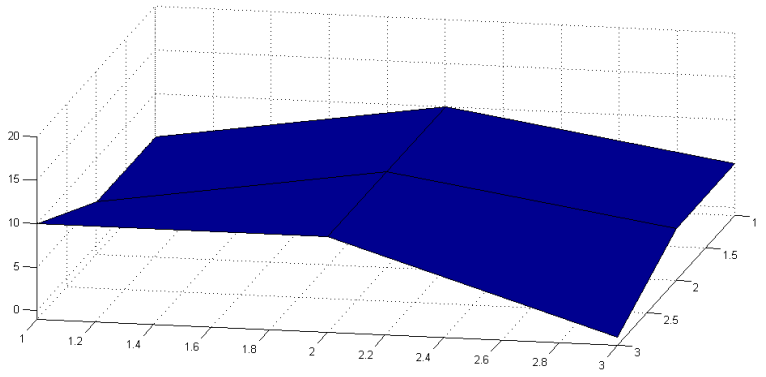
(a) X error



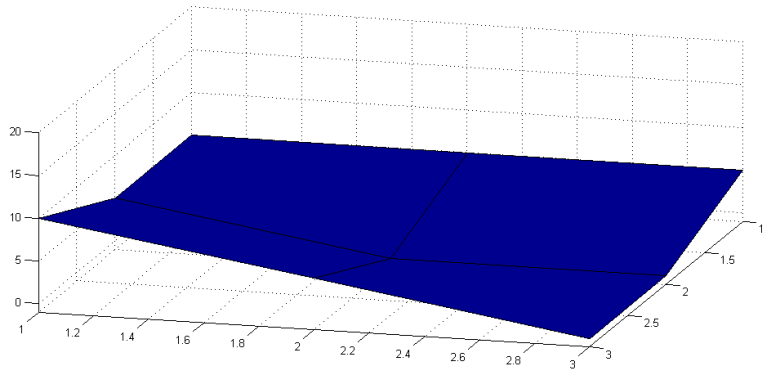
(b) Y error

Figure 3.14: Error with plane set at 30 mm from screen frame.

to give a notable degradation in quality. Again, horizontal errors were consistent with previous plane's, around 0-5mm at +10mm from the plane, 5-15mm when +20mm from the plane and 20-40mm at +30mm from the plane. At this distance from the screen it also became practical to start measuring error when deviating towards the screen. These errors proved less significant than deviating away from the screen, with errors of 10-30mm in the horizontal direction and 5-10 mm in the vertical plane at -30mm from the plane.



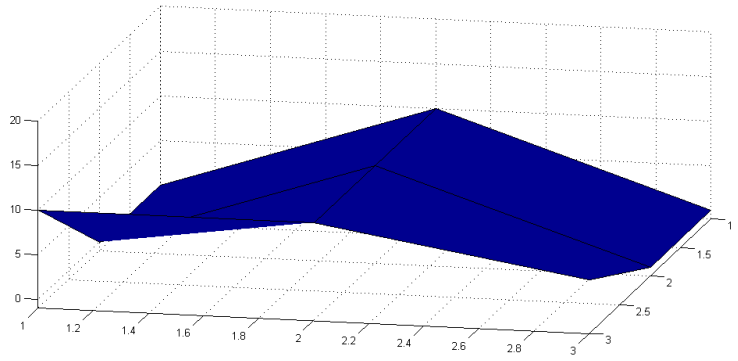
(a) X error



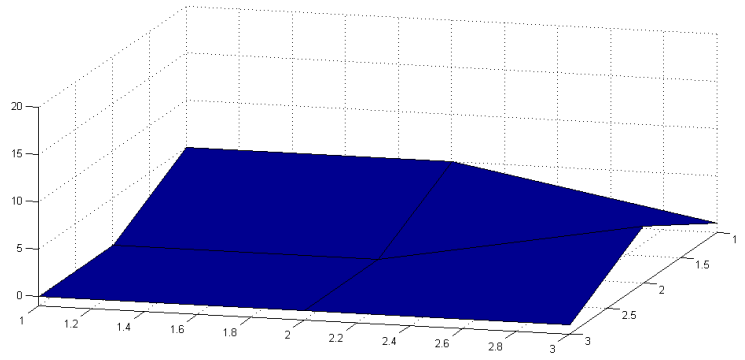
(b) Y error

Figure 3.15: Error with plane set at 40 mm from screen frame.

At 40mm (3.15) more noticeable error starts to creep in. This is likely a result of the fact that the plane occupies a smaller region of the cameras field of view as the change in distance on the far side is smaller in pixels than the close side. This error can be compensated for somewhat with higher resolution cameras, giving greater fidelity over the same distances. When deviating from the plane, errors from 5mm to 10mm are experienced up to 20mm in or out from the plane. At +30mm there is 20-30mm of error in the x direction and 10-20mm in the y direction. At -30mm there is a horizontal error of 10-30mm, and 5-10mm vertically.



(a)



(b)

Figure 3.16: Error with plane set at 50 mm from screen frame.

Finally, at 50mm (3.16) a similar amount of error is shown as at 40mm. The lower resolution hampers measurements at the far side from the camera and, more importantly, calibration errors take effect. The further from the screen the plane gets, the bigger the error introduced by the calibration process as it becomes more difficult for the user to maintain a steady distance from the screen and keep their hands precisely lined up with the corners during calibration. When deviating in from the plane there is very little vertical error, with 5-10mm at -30mm and

a more noticeable 10-30mm horizontally. When moving away from the screen the horizontal error builds steadily, with 5mm at +10mm, 10-20mm at +20mm and at +30mm, 20-40mm of error. Vertically the error is less than 10mm until a deviation of +30mm is reached from the plane, where error is around 10-20mm.

These results show that error is fairly minimal when not deviating significantly from the plane of calibration, with a maximum error of 10mm at 50mm from the screen border. This can mostly be attributed to the lack of resolution in the cameras for their distance from the screen, and to directional biases introduced by the asymmetry of the camera setup necessitated by the decreased field of view of the right camera. By placing the cameras closer to the mid level of the screen the overall horizontal errors would be reduced as the distance to deviate from this point will be smaller (half the screen), minimizing the effects of calibration inaccuracies.

User Testing

Because of the user-centric intent of this system it was decided that the most appropriate method of testing it's effectiveness

Testing of the system was carried out using subjective user evaluation. The evaluation was in the form of a brief demonstration of the systems operations, followed by having the user complete a series of tasks using the system and evaluating their ease on a questionnaire (See Appendix). The system was setup as in figure above 3.10, and users were not previously familiar with the system or its operation.

Through testing one thing that became immediately apparent was that users

had difficulty calibrating the system without guidance. This was partially due to a lack of onscreen prompts (something that could easily be added to the UI) at start up, and because many did not initially understand the concept of defining the plane of use and how it related to the systems operation (This was due to a failure of communication during the test briefing, and a lack of familiarity with the project). Despite these difficulties the simplicity of initialisation was rated at an average of 7.14 on a scale of 1 to 9 (1 being low scoring, 9 being high scoring).

The tasks of hovering over the large box and the small button on the screen were given a simplicity rating of 8.38 and 8.41 respectively (although one user did not submit a score for hovering over the large box) The magnitude of these scores is most likely attributable to user interface design conventions in touch screen application design. The box area and the circle were both of reasonable size to due to the inherent inaccuracy of touch screen systems (as discussed in Section 1.1), and this design feature also acted to increase the usability of the system. Interestingly, despite the fact that hovering over the small button should have been a more difficult task, both tasks were given the same ratings by users with the exception of there being one less rating of 9 for the large box selection. This may possibly be attributed to the missing vote for that task from one user, although this cannot be certain.

The zooming in and out gesture task received the same average rating as the calibration task, however it was not as evenly distributed (all previous tasks were approximately normally distributed). Some found the task easy (rating of 9), while others found the task more difficult (ratings of 6-7), displaying a clear spread between the two. One possible reason for this spread is that the accuracy of the gesture is highly dependant on the calibration phase. If the user does not

correctly specify the plane they intend to use the screen in then any depth calculations will not return the results they expect. For example, if a user specified the plane of use to be one inch from the screen an parallel with it but then proceeded to use the screen slightly off parallel, the depth value reported would be inconsistent from one side of the screen to the other as the user deviated from the plane they specified to follow the plane they expected. This can be rectified by ensuring the user understands the calibration task better and thus, can perform it more to their preference.

The final task given to users was to experiment with using the system, and to rate the system based on accuracy of the input, performance (in terms of frame rate), and the overall user opinion of the system for input. As expected, performance was rated the lowest of these factors, with an average rating of 6.36 on a scale of 1 to 9. This was due to the lack of optimisations of the test system, and there is plenty of room for both software optimisations, and hardware acceleration (porting to CUDA/OpenCL or using SSE instructions etc.) to improve performance to much higher levels. Interestingly, the user rating of accuracy was lower than other results would have indicated at an average of 6.61, despite receiving higher ratings for the ease of selecting objects and performing the zoom gesture. This is likely due to the difficulties unfamiliar users faced with the limitations of the current point of interest detection system as discussed in Section 4.2.4. The last thing users were asked for after their free trial period was to give an overall rating of how well the system worked. Overall users gave the system a rating of 7.32 with most of the votes being either 7 or 8.

These results demonstrate that from a user perspective, the solution is adequate for the task of providing input to a human-computer interface. There is room

for improvement in the area of performance to improve the user experience, and further educational material should be provided to users to ensure correct use in future. Lastly, while user education can circumvent the issue of incorrect use degrading accuracy of the system, finding an alternative method of calculating where the user is pointing to may be a worth while step as it mitigates this problem.

3.4 Problems Encountered

As with any image processing based problem, the environment was one of the biggest obstacles faced in this project. In order to create a robust solution that is capable of operating in any environment there are two approaches that can be followed: One can anticipate all undesirable conditions that may arise, and attempt to include functionality that will negate these undesirable affects; Alternatively, one may include functionality within the system that mitigates the undesirable conditions, thus rendering them irrelevant. While simply controlling ones environment by mitigating any possible environmental issues if the most convenient to implement from a development perspective, controlling ones environment is not always practical from a user perspective.

Weak ambient lighting, or coloured ambient lighting can cause many issues with colour detection. Weak lighting leads to colours being represented as much darker, and increases the difficulty of detection, as it compresses all colours to the darker end of the spectrum, where skin colour is less defined and harder to separate from non-skin tones. This undesirable behaviour can either be mitigated by ensuring the environment of use is always strongly lit with white light, or by taking lighting into account and normalising every captured frame based on a reference lighting level and colour.

Strong lighting from an LCD screen can also cause colour tinting of the hand when it is close to the screen, leading to incorrect detection as a non-skin tone. This is obviously highly undesirable, and needs to be mitigated (negating this is too impractical as it requires dimming the users screen to very low levels, making it difficult for the user to see what is on the screen properly). The best way to achieve this is to ensure strong, white ambient lighting as this overpowers the light source of the screen and ensures correct colour detection by the cameras. Another step that can be taken to mitigate this issue is to design user interfaces in such a manner that they do not present strong coloured lighting by maintaining a high level of near white content. In the scenario of a kiosk contamination from the screen is less of a problem as the kiosk enclosure can be designed specifically to minimise contamination from screen lighting.

A final issue faced was the requirement for the user to hold their hand in an optimum orientation to get the best results. This was highly dependant on camera positioning and may not be practical in space limited situations like when the bottom of the screen is near to the desk leaving insufficient room for proper orientation.

Chapter 4

Final Thoughts

4.1 Conclusions

Through my work on building a 'Near Touch Interface' system and developing the prototype soccer demonstration system, the viability of the proposed system and it's capability of delivering what was required of it have been demonstrated. The system has been able to detect the position of the user's hand with sufficient accuracy, as shown by user testing of the system. This ability can be further improved through further testing and development to become more accurate and reliable. The systems ability to recognise simple gestures was demonstrated both in the user testing of the system (for the zoom gesture), and in the soccer demonstration (as it used a similar data input and output, and was successful in detecting a kicking motion accurately enough to allow power and angle control).

With the demonstration of the systems viability, I believe there is grounds for further development and improvement on the developed system to help optimise and polish it further, to the point where it becomes a viable commercial application.

4.2 Accomplishment of Goals

4.2.1 Demonstrate the ability to track the users hand in front of the screen

This was successfully demonstrated by the final system in its current state and clearly visualised by the system. Video feedback can be seen from both cameras on the display, and the scene is shown as a binary image, where the skin detected is shown as white, and background is black. The largest object of interest (the user's hand) is further segregated and displayed with a bounding box to highlight its location. The point the user is pointing too is taken to be the top corner of this bounding box(left or right, depending on which side the camera is on). While this produces satisfactory results, it requires the user hold their hand at a certain orientation to the screen for optimum results. A method based on skeletonisation of the recognised hand object could result in a more accurate idea of where the finger points too, as well as the possibility of multiple cursors using more fingers.

4.2.2 Display Ability to report the point they are pointing too with sufficient accuracy for general use

The accuracy of the system was tested subjectively through user testing and feedback. Provided correct calibration of the system and an appropriate environment of use, accuracy was found to be acceptable by users. The full screen can be accessed provided the calibrated plane is not too far from the screen, and items within can be selected with ease. The depth functionality, while simplified proved reasonably simple for new users to successfully use as long as the calibration plane was not too close to the screen for the given threshold. For accuracy, the biggest factor in reducing the users ability to accurately use the system was the calibration phase. This proved the most difficult to properly achieve, and the accuracy of the system relies on accurate specification of the plane of detec-

tion. This may have been due to unfamiliarity with the system and a deficiency in proper instruction as the users were given only a brief demonstration of the process to instruct, and were not previously familiar with the systems workings. This was done to ensure they did not have any preconceived opinions about the system, so they would remain unbiased when evaluating it.

While noise will likely make reliable pixel point accuracy impractical to achieve, this is nullified by user interface design conventions for touch screen interfaces necessitated by the imprecise nature of touch (the point of touch is the whole finger tip, which can be a very large area if the user presses down hard). Accuracy could also be made more robust with regards to deviation from the use plane using stereoscopy to map the user's hand to true 3D space as opposed to 2D space with an approximation of depth.

4.2.3 Recognise simple gestures

It was a goal of the system to have a simple gesture recognition system to facilitate common actions such as back and forwards and scrolling when navigating web browsers or documents. This was demonstrated somewhat in the kicking motion detected by the demonstration was a good approximation of simple gesture recognition in the final system, as it detects both a simple motion and infers meaningful information from this gesture, such as a direction and magnitude. Given the similarities between the output of the detection algorithms between the soccer demonstration and the final near touch system, this is a good indicator that simple gesture recognition would have worked well in the final solution. The 'zoom' gesture was also implemented within the prototype user interface, showing that it was a sufficiently accurate and viable gesture through user testing and feedback.

4.2.4 Display this in action using a graphical user interface

This goal was met, but not to the standard originally aimed for. It was originally planned for the system to run over the top of windows, or to emulate a windows like system to showcase the potential application, but in the interest of time and practicality, a more simple interface was implemented for user testing and to demonstrate proof of the concepts feasibility. This developed interface successfully showcased the systems ability to handle simple zoom in/out functionality, as well as accurately track hand movement sufficiently to interact with typical touch screen sized buttons.

4.3 Future Work

While the system proved to be suitable for use according to user testing, there was room to improve both the accuracy of the system, and especially the performance. Performance can be fixed through optimisation of the various methods implemented. Some suggestions for improving the accuracy and reliability of the system are:

4.3.1 Complete Hand Detection

Currently the system relies simply on detection of skin tones and a simple bounding box approximation to determine where the user is pointing. While this proves sufficiently robust for basic use in most cases (and the added benefit of not requiring a perfectly formed hand being the only possible input method) , ideally a more robust solution should be used to cover edge cases. Possible hand detection methods such as Mase and Suenaga's Pointing detection algorithm [47] or something similar to Mysliwiec's FingerMouse hand detection algorithm [48]. Alternatively, detection could be handled using a combination of Corner detection such as Trajkovic-Hedly [49] or SUSAN [50] and Gesture recognition techniques

such as Hidden Markov Models. While this puts a constraint on how the user may interact with the system, it also allows for much more accurate detection of point gestures in particular, leading to more accurate input.

4.3.2 Migrate to True Stereoscopy

A move to true stereoscopy for detection of the hands location would be largely beneficial to accuracy, as well as adding much more accurate information about z-depth. This would only be practical for use in controlled environments such as advertising kiosks due to a reliance on knowledge of the cameras position (this cannot reasonably be controlled in a home user environment that relies on the user placing the cameras themselves). On a controlled advertising kiosk where the camera placement will be known stereoscopy provides much more reliable accuracy, as well as more information about the hand position. This will enable the implementation of much more complex gestures, as well as granting much greater accuracy to the user, especially in gestural control along the z-axis. It may be possible to gather enough information to determine all necessary parameters without having exact knowledge of camera placement. This could be achieved using calibration against an object of known size for instance.

4.3.3 Colour temperature correction

In any image processing problem the variable that has the highest impact is always lighting of the environment. It is highly desirable to be able to control lighting as it can vastly simplify the problem to be solved, reducing both work required to achieve the task and computational cost. Controlling lighting however is an almost completely unreasonable requirement for a general system to be used in commercial environments such as retail stores, or in home environments for typical home users. In commercial situations such as kiosks for shops certain

control elements can be reasonably introduced via a partnership between the software creator and a hardware manufacturer that creates touch screen based retail kiosks etc.

This can be achieved by creating an integrated system using a hardware unit with controlled shading and lighting internally, and appropriate software. There is still, however the problem of external lighting which cannot be controlled (forcing all stores to conform to a specific set of lighting requirements is unreasonable and impractical) and cannot be mitigated without creating a fully enclosed booth (something that would make a display to be used by multiple people in quick succession fairly impractical). This is also a problem that cannot be reasonably controlled in a home use environment. A practical solution to this conundrum is to implement appropriate software control over the lighting by measuring its colour temperature and correcting for discrepancies. This also is desirable as most affordable web camera systems implement their own colour correction, leading to an image that does not adequately represent the state of reality.

To remove the effect of the lighting and camera introduced colour temperature, a pure white(♯ FFFFFF RGB) screen can be displayed, and then compared with the displayed image from the cameras. The difference between pure white and what is seen by the cameras is the effect of the lighting and camera on colour temperature, and thus can be removed by taking this offset off each image. This correction will leave a lighting and camera independent image for use with the system, which allows for much tighter tolerances on the ranges for skin tone detection etc.

4.3.4 Auto skin colour calibration

Reliable skin detection generally requires a range of colour values to be used to work for the wide range of skin tones presented by the human race, from very dark browns to pale pinkish reds. Because of this range of skin tones to be searched over there are often overlaps with the colours of other objects, most commonly wood tones, and this can lead to high rates of false positives in many environments. To compensate for this the search space can be minimised by registering a specific user's skin tone, and narrowing the search space to only include the users approximate colour range.

This can be implemented as part of an initialisation routine that takes a static image from the camera of the users hand over a plain white background. From this the background can be removed and an average colour gained from the resulting image (ignoring non skin tones to remove noise from articles such as clothing and jewellery). This provides an almost completely automated system that requires a minimum of user co-operation, and can be run quickly at any time. To further improve the accuracy of the skin detection it could also be implemented that the user can select from the image a bounding area for their skin so that only skin tones are considered. This however requires more time and user interaction, making it less practical for systems such as kiosks which require many vastly different users to be able to immediately begin using the system with no problems for short periods of time before switching to a new user.

Alternatively, methods such as the Yang et al. arbitrary colour classifier[51] could be employed.

Appendix A

Questionnaire

Near Touch User Interface Test Evaluation:

Task 1: Initialising the System

Please initialise the system as demonstrated

How simple was the system to initialise on a scale of 1 to 9, with 1 being extremely difficult and 9 being extremely simple?

1.....2.....3.....4.....5.....6.....7.....8.....9

Task 2: Hover over the big box in the centre of the screen

Please hover the cursor over the big box in the centre of the screen until it goes green

How simple was it to complete this task on a scale of 1 to 9, with 1 being extremely difficult and 9 being extremely simple?

1.....2.....3.....4.....5.....6.....7.....8.....9

Task 3: Hover over the small button in the lower right of the screen

Please hover the cursor over the small button in the lower right of the screen until it goes green

How simple was it to complete this task on a scale of 1 to 9, with 1 being extremely difficult and 9 being extremely simple?

1.....2.....3.....4.....5.....6.....7.....8.....9

Task 4: Make the box in the centre of the screen change colours

1. **Make the box go yellow by moving closer to the screen**

2. Make the box go purple by moving further from the screen

How simple was it to complete this task on a scale of 1 to 9, with 1 being extremely difficult and 9 being extremely simple?

1.....2.....3.....4.....5.....6.....7.....8.....9

Task 5: Play around for a little bit

Please rate the accuracy of the system on a scale from 1 to 9, with 1 being highly inaccurate and 9 being pinpoint accuracy.

1.....2.....3.....4.....5.....6.....7.....8.....9

Please rate the performance of the system on a scale of 1 to 9, with 1 being unusably slow and 9 being more than fast enough.

1.....2.....3.....4.....5.....6.....7.....8.....9

Please rate the overall user experience using the system on a scale from 1 to 9, with 1 being unusable and frustrating and 9 being enjoyable and simple to use.

1.....2.....3.....4.....5.....6.....7.....8.....9

Other Comments:

References

[1] "Wii at nintendo, last visited 4/05/10." http://www.nintendo.com/wii/.

[2] "Microsoft official press release on natal, last visited 22/07/09.." http://www.microsoft.com/presspass/press/2009/jun09/06-01e3pr.mspx.

[3] "Studio one 19 desktop, last visited on 3/6/10." http://www1.ap.dell.com/nz/en/home/desktops/desktop-studio-one-19/pd.aspx?refid=desktop-studio-one-19&s=dhs&cs=nzdhs1.

[4] "Apple iphone technical specifications, last visited 4/05/10." http://www.apple.com/iphone/specs.html.

[5] "Apple ipod touch technical specifications, last visited 4/05/10." http://www.apple.com/ipodtouch/specs.html.

[6] "Apple ipad technical specifications, last visited 4/05/10." http://www.apple.com/ipad/specs/.

[7] "Microsoft surface technical resources, last visited 4/05/10." http://www.microsoft.com/surface/en/us/Pages/Technical/Learn.aspx.

[8] "Nexus one phone - features overview & technical specifications, last visited 4/05/10." http://www.google.com/phone/static/en_US-nexusone_tech_specs.html.

[9] "Windows 7 development team blog on win7 touch development, visited 27/07/09.." http://windowsteamblog.com/blogs/windows7/archive/2009/05/27/introducing-the-microsoft-touch-pack-for-windows-7.aspx.

[10] "Windows phone 7 design system - codename metro, last visited on 3/2/12." http://go.microsoft.com/fwlink/?LinkID=189338.

[11] J. A. Totty, "Mid-band public multimedia kiosks," *BT Technology Journal*, vol. 13, pp. 97–104, 1995.

[12] "Razer products page for lachesis gaming mouse, last visited 22/07/09.." http://www.razerzone.com/gaming-mice/razer-lachesis/.

[13] "Unlimited realities fingertapps site, last visited 23/07/09." http://www.fingertapps.com/.

[14] "Unlimited realities dell touchzone, last visited 6/05/10." http://www.ur.co.nz/default.asp?pageid=13.

[15] "United states department of justice, case: United states vs microsoft, last visited 22/07/09.." http://www.usdoj.gov/atr/cases/ms_index.htm.

[16] "Nintendo product page for wii motion plus, last visited 22/07/09.." http://www.nintendo.com/wii/what/accessories/wiimotionplus.

[17] "Prime sense product specifications, last visited on 25/05/10." http://www.primesense.com/files/PDF/PrimeSensor_RD1.08_Datasheet.PDF.

[18] "Peter molyneaux's presentation during microsoft xbox 360 press conference at e3 2009 as covered by kotaku press, last visited 22/07/09.." http://www.kotaku.com/5274554/molyneuxs-milo-brings-a-virtual-child-to-the-xbox-360.

[19] "Gamasutra interview with 3dv systems on their z-cam technology, last visited 22/07/09.." http://www.gamasutra.com/php-bin/news_index.php?story=16637.

[20] "Playstation america press release on pseye, last visited 22/07/09.." http://www.us.playstation.com/news/pressreleases/396.

[21] "Zdnet news coverage on iran ps2 bomb control fears, last visited 22/07/09.." http://news.zdnet.co.uk/hardware/0,1000000091,2083302,00.htm.

[22] "Alex popovich's blog, last visited 22/07/09.." http://alexpopovich.wordpress.com/.

[23] "Don't click it!, last visited 11/05/10." http://www.dontclick.it.

[24] A. H. F. Lam and W. J. Li, "Mids: Gui and tui in mid-air using mems sensors," in *International Conference on Control and Automation*, pp. 1218–1222, 2002.

[25] J. Segen and S. Kumar, "Gesture vr: vision-based 3d hand interace for spatial interaction," in *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, pp. 455–464, 1998.

[26] G. Shoemaker, L. Findlater, J. Q. Dawson, and K. S. Booth, "Mid-air text input techniques for very large wall displays," in *Proceedings of Graphics Interface*, pp. 231–238, 2009.

[27] C.-M. Karat, C. Halverson, D. Horn, and J. Karat, "Patterns of entry and correction in large vocabulary continuous speech recognition systems," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 568–575, 1999.

[28] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a wii controller," in *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pp. 11–14, 2008.

[29] T. Shiratori and J. K. Hodgins, "Accelerometer-based user interfaces for the control of a physically simulated character," *ACM Trans. Graph.*, vol. 27, pp. 1–9, 2008.

[30] A. D. Wilson, "Touchlight: an imaging touch screen and display for gesture-based interaction," in *Proceedings of the 6th international conference on Multimodal interfaces*, pp. 69–76, 2004.

[31] T. G. Myron W. Krueger and K. Hinrichsen, "Videoplace - an artificial reality," in *ACM SIGCHI Bulletin archive*, vol. 16, pp. 35–40, 1985.

[32] J. C. Lee, "Hacking the nintendo wii remote," *Pervasive Computing, IEEE*, vol. 7, no. 3, pp. 39–45, 2008.

[33] S. Elrod, R. Bruce, R. Gold, D. Goldberg, F. Halasz, W. Janssen, D. Lee, K. McCall, E. Pedersen, K. Pier, J. Tang, and B. Welch, "Liveboard: A large interactive display supporting group meetings, presentations, and remote collaboration," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 599–607, 1992.

[34] M. Czernuszenko, D. Pape, D. Sandin, T. DeFanti, G. L. Dawe, and M. D. Brown, "The immersadesk and infinity wall projection-based virtual reality displays," *SIGGRAPH Comput. Graph.*, vol. 31, pp. 46–49, 1997.

[35] R. Yang, D. Gotz, J. Hensley, H. Towles, and M. S. Brown, "Pixelflex: A reconfigurable multi-projector display system," in *Proceedings of the Conference on Visualization*, pp. 167–174, 2001.

[36] M. Ringel, H. Berg, Y. Jin, and T. Winograd, "Barehands: Implement-free interaction with a wall-mounted display," in *Extended Abstracts on Human Factors in Computing Systems*, pp. 367–368, 2001.

[37] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer, "Easyliving: Technologies for intelligent environments," in *Handheld and Ubiquitous Computing*, Lecture Notes in Computer Science, pp. 97–119, 2000.

[38] J. Wobbrock, A. Wilson, and Y. Li, "Gesture without libraries, toolkits or training: A $1 recogniser for user interface prototypes," in *20th annual ACM symposium on User interface software and technology*, pp. 159–168, 2007.

[39] M. Piccardi, "Background subtraction techniques: a review," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 4, pp. 3099–3104, Oct. 2004.

[40] F. Porikli and O. Tuzel, "Human body tracking by adaptive background models and mean shift analysis," in *Conference on Computer Vision Systems, Workshop on PETS, IEEE*, 2003.

[41] D. Marius, S. Pennathur, and K. Rose, "Face detection using colour thresholding, and eigenimage template matching." `http://scien.stanford.edu/pages/labsite/2003/ee368/Project/reports/ee368group15.pdf`, 2003. Stanford University Project, last visited on 19/3/2012.

[42] "Opencv sourceforge development page, last visited on 24/07/09." `http://sourceforge.net/projects/opencv/`.

[43] "Intel opencv integrated performance primitives faq, last visited on 24/07/09." `http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-intel-ipp-open-source-computer-vision-library-opencv-faq/`.

[44] "nvidia cuda website, last visited 24/07/09." `http://www.nvidia.com/object/cuda_home.html#`.

[45] "Opencl homepage, last visited 24/07/09." `http://www.khronos.org/opencl/`.

[46] "Unlimited realities at cebit, last visited on 26/05/10." `http://www.ur.co.nz/default.asp?pageid=9&data_articleID=43&returnid=1`.

[47] M. Mase and K. Suenaga, "Real-time detection of pointing actions for a glove-free interface," in *In IAPR Workshop on Machine Vision Applications*, pp. 473–476, 1992.

[48] T. Mysliwiec, "Fingermouse: A freehand computer pointing interface," in *in Proc. of Intl Conf. on Automatic Face and Gesture Recognition*, pp. 372–377, 1994.

[49] M. Trajkovic and M. Hedley, "Fast corner detection," *Image and Vision Computing*, vol. 16, no. 2, pp. 75 – 87, 1998.

[50] S. M. Smith and J. M. Brady, "Susan - a new approach to low level image processing," *International Jounal of Computer vision*, vol. 23, no. 1, pp. 45–78, 1997.

[51] X. Zhu, J. Yang, and A. Waibel, "Segmenting hands of arbitrary color," in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pp. 446–453, 2000.