

Visual Based Finger Interactions for Mobile Phones

A thesis submitted in fulfillment
of the requirements for the degree of

MASTER OF SCIENCE

of

RHODES UNIVERSITY

by

Simon Kerr

December 2009

Abstract

Vision based technology such as motion detection has long been limited to the domain of powerful processor intensive systems such as desktop PCs and specialist hardware solutions. With the advent of much faster mobile phone processors and memory, a plethora of feature rich software and hardware is being deployed onto the mobile platform, most notably onto high powered devices called smart phones. Interaction interfaces such as touchscreens allow for improved usability but obscure the phone's screen. Since the majority of smart phones are equipped with cameras, it has become feasible to combine their powerful processors, large memory capacity and the camera to support new ways of interacting with the phone which do not obscure the screen. However, it is not clear whether or not these processor intensive visual interactions can in fact be run at an acceptable speed on current mobile handsets or whether they will offer the user a better experience than the current number pad and direction keys present on the majority of mobile phones. A vision based finger interaction technique is proposed which uses the back of device camera to track the user's finger. This allows the user to interact with the mobile phone with mouse based movements, gestures and steering based interactions. A simple colour thresholding algorithm was implemented in Java, Python and C++. Various benchmarks and tests conducted on a Nokia N95 smart phone revealed that on current hardware and with current programming environments only native C++ yields results plausible for real time interactions (a key requirement for vision based interactions). It is also shown that different lighting levels and background environments affects the accuracy of the system with background and finger contrast playing a large role. Finally a user study was conducted to ascertain the overall user's satisfaction between keypad interactions and the finger interaction techniques concluding that the new finger interaction technique is well suited to steering based interactions and in time, mouse style movements. Simple navigation is better suited to the directional keypad.

Contents

CHAPTER 1 - INTRODUCTION	1
1.1. INTRODUCTION	1
1.2. PROBLEM STATEMENT	3
1.3. SCOPE OF RESEARCH	4
1.4. RESEARCH METHODOLOGY	4
1.5. SUMMARY OF FINDINGS	5
1.6. THESIS ORGANISATION	5
CHAPTER 2 - RELATED WORK	7
2.1. INTRODUCTION	7
2.2. MULTIMODAL INTERFACES	8
2.2.1. <i>TYCOON</i>	8
2.2.1.1. Transfer	9
2.2.1.2. Equivalence.....	10
2.2.1.3. Specialisation	10
2.2.1.4. Redundancy	10
2.2.1.5. Complementarity	10
2.2.1.1. Relation to Finger Interaction Technique	11
2.2.2. <i>Multimodal Gesture-Based Interfaces</i>	11
2.2.3. <i>Gestures Relating to Finger Interactions</i>	12
2.3. COMPUTER VISION BASED TECHNOLOGIES	15
2.3.1. <i>Desktop and PDA Technologies</i>	15
2.3.2. <i>Image Manipulation</i>	17
2.3.3. <i>Motion Detection</i>	17
2.3.4. <i>Taxonomy of Input Devices</i>	20
2.3.5. <i>Mobile Interfaces</i>	22
2.4. ALGORITHMS	24
2.4.1. <i>Motion Detection</i>	24
2.4.2. <i>Thresholding</i>	25
2.4.3. <i>Optical Flow</i>	25
2.4.4. <i>Types of algorithms</i>	26
2.5. ADVANCES IN MOBILE TECHNOLOGY	27

PRELIMINARIES

2.6.	SOFTWARE INTERFACES	28
2.6.1.	<i>NokiaCV</i>	28
2.6.2.	<i>Java MIDP and the MMAPi</i>	29
2.6.3.	<i>Microsoft .NET Compact Framework</i>	29
2.6.4.	<i>Python S60</i>	30
2.7.	SUMMARY	30
CHAPTER 3 - ALGORITHM OVERVIEW.....		31
3.1.	INTRODUCTION	31
3.2.	PROPOSAL OF ALGORITHM	34
3.3.	OVERVIEW OF ALGORITHM	34
3.3.1.	<i>Camera</i>	35
3.3.2.	<i>Input</i>	35
3.3.3.	<i>Calibration</i>	36
3.3.4.	<i>Tracking</i>	36
3.3.5.	<i>Search Square</i>	39
3.3.6.	<i>Output</i>	40
3.4.	SUMMARY	41
CHAPTER 4 - PERFORMANCE ANALYSIS		42
4.1.	INTRODUCTION	42
4.2.	METHODOLOGY.....	43
4.2.1.	<i>Platform</i>	45
4.2.2.	<i>APIs</i>	45
4.2.3.	<i>Algorithm</i>	46
4.3.	RESULTS AND DISCUSSION	47
4.3.1.	<i>FPS</i>	48
4.3.2.	<i>Time to Track</i>	48
4.3.3.	<i>Full Scan</i>	48
4.3.4.	<i>Pixel Skip</i>	50
4.3.5.	<i>Latency</i>	51
4.3.6.	<i>Speed and Deployability</i>	51
4.4.	SUMMARY	53
CHAPTER 5 - ACCURACY, COLOUR AND LIGHTING		54

PRELIMINARIES

5.1. INTRODUCTION54

5.2. METHODOLOGY.....56

 5.2.1. *Light Level Study*56

 5.2.2. *Backgrounds*58

 5.2.3. *Accuracy Measurements*.....59

 5.2.4. *Special Cases*.....61

5.3. RESULTS61

 5.3.2. *Analysis*.....64

 5.4.1.1. *Bright*.....64

5.4. SUMMARY66

CHAPTER 6 - USER STUDY68

6.1. INTRODUCTION68

6.2. PERFORMANCE METRICS69

 6.2.2. *Time-on-Task*.....70

 6.2.3. *Errors*.....70

6.3. TASKS.....70

 6.3.1. *Steering game*71

 6.3.2. *Target Acquisition*72

 6.3.3. *Directional Gestures*.....75

6.4. METHODOLOGY.....76

 6.4.1. *Apparatus*.....77

 6.4.2. *Participants*79

 6.4.3. *Procedure*79

6.5. RESULTS AND DISCUSSION81

 6.5.1. *Time on Task*.....81

 6.5.2. *Overall Errors*82

 6.5.3. *TLX Score*83

 6.5.4. *Preferred Interaction technique*.....84

 6.5.5. *Target Acquisition*85

 6.5.6. *Steering Game*88

 6.5.7. *Directional Gestures*.....89

 6.5.8. *USE/SUS questionnaire*.....91

6.6. SUMMARY.....92

PRELIMINARIES

CHAPTER 7 - DISCUSSION	94
7.1. DISCUSSION	94
7.1.1.1. Faster Interactions	97
7.1.1.2. Recognition	98
7.1.2. <i>Battery Life</i>	98
7.1.3. <i>Deploying the Finger Interaction Technique</i>	99
7.2. SUMMARY	101
CHAPTER 8 - CONCLUSION AND FUTURE WORK	102
8.1. FUTURE WORK	103
7.3.1. <i>Future Work</i>	<i>Error! Bookmark not defined.</i>
References	106
Appendices	115
APPENDIX A	115
APPENDIX B	126
APPENDIX C	137

PRELIMINARIES

List of Tables

Table 3.1: Stream colour.....	37
Table 4.1: Speed Results	47
Table 5.1: Motion Capture Systems.....	55
Table 5.2: Average light values in light study.....	58
Table 5.3: Accuracy Ratings.....	62
Table 6.1: Latin Square Design.....	77
Table 7.1: Foley et al. sub tasks.....	97

List of Figures

Figure 2.1: TYCOON classifications.....	9
Figure 2.2: Back-of-device input.....	11
Figure. 2.3: Convergence of mobile technologies.....	16
Figure 2.4: Sobel Filter example.....	18
Figure 2.5: Marker based tracking.....	19
Figure 2.6: Kick-up menus.....	20
Figure 2.7: Motion detection game examples.....	22
Figure 2.8: <i>Finteraction</i> one hand control method.....	23
Figure 2.9: Augmented reality examples.....	24
Figure 2.10: Optical Flow Field.....	26
Figure 2.11: Progression of Nokia interfaces type and screen size.....	28
Figure 3.1: Single handed interaction technique.....	32
Figure 3.2: Thresholding example.....	32
Figure 3.3: Search Square.....	33
Figure 3.4: Continuous pixels example.....	38
Figure 3.5: Pixel stream and search square.....	40
Figure 4.1: Frames per second for the three APIs.....	49
Figure 4.2: Pixel skipping results.....	51

PRELIMINARIES

Figure 4.3: API's deployability and speed.....	52
Figure 5.1: Complex, textured and simple scenes.....	58
Figure 5.2: Anti-alias effect.....	59
Figure 5.3: Ranking System.....	60
Figure 5.4: Accuracy calculation.....	61
Figure 5.5: Special Cases.....	63
Figure 5.6: Bright light level	65
Figure 5.7: Finger on similar colour background.....	65
Figure 5.8: Complex scene under two different lighting conditions.....	66
Figure 6.1: User playing the steering game.....	71
Figure 6.2: User using target acquisition application.....	73
Figure 6.3: Finger path in target acquisition task.....	74
Figure 6.4: Reattempt at selecting a block.....	74
Figure 6.5: User using directional gestures application.....	75
Figure 6.6: Overshoot error in directional gestures task.....	76
Figure 6.7: Task time.....	82
Figure 6.8: Errors.....	83
Figure 6.9: Average mean score.....	84
Figure 6.10: The amount of participants who preferred one interaction technique over the other.....	85
Figure 6.11: Target acquisition subcategory scores.....	87
Figure 6.12: Steering game subcategory scores.....	88
Figure 6.13: Directional gestures subcategory scores.....	90
Figure 6.14: USE/SUS questionnaire results.....	91
Figure 8.1: Grid Block Matching.....	104

List of Listings

Listing 3.1: Multiband thresholding algorithm	35
Listing 3.2: Initial colour assignment and core tracking algorithm.....	39

PRELIMINARIES

Listing 4.1: Obtaining a bitmap frame using NokiaCV46

PRELIMINARIES

Acknowledgements

I would like to thank my supervisors Greg Foster and Hannah Thinyane both were fantastic to work with and both have taught me much in the past two years. I would also like to thank my family for all their support and love.

Finally I would like to thank Peter Clayton and the NRF for the bursary they gave me as well as the Centre of Excellence and their sponsors for the phones, without which I could not have done this study.

PRELIMINARIES

Publications

Chapters 2 and 4 were the basis of two published conference papers. The full paper *Vision Based Interaction Techniques for Mobile Phones: Current Status and Future Directions* which was based on Chapter 2 was published at SATNAC 2008.

Kerr, S., Thinyane, H. and Foster, G. (2009). *Mobile Phone Performance Analysis for Camera Based Visual Interactions* . Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, Sasolburg.

The full paper *Mobile Phone Performance Analysis for Camera Based Visual Interactions* which was based on Chapter 4 was published at SAICSIT 2009 and attracted the award of runner-up for best paper at the conference.

Kerr, S., Foster, G. and Thinyane, H. (2008). *Vision Based Interaction Techniques for Mobile Phones: Current Status and Future Directions* . Southern Africa Telecommunication Networks and Applications Conference '08. Durban.

The abstracts for these papers can be found in Appendix C.

Chapter 1 - Introduction

The current way of interacting with mobile phones is closely linked to the limitations of the current keypad layout (Wisniewski, et al., 2005). The major factor limiting the use of more precise, desktop-style interactions has always been the small physical size of handsets and digital buttons on the keypad of mobile phones (Forman & Zahorjan, 1994). This thesis will propose a new finger-tracking based interaction technique which will involve using the available multimedia technologies integrated in the mobile phone unit, namely the camera. The camera will act as the input for back-of-device touchscreen style interactions. An implementation of this camera based interaction system using a finger as the main interaction tool and usability and user satisfaction are the final deliverables of this study. The implementation allows the user to interact with applications and games using their fingers, creating a more rich and versatile interaction experience for certain types of interaction.

1.1. Introduction

Mobile phones have become a standard form of communication and mobile computing in much of the world. Current penetration worldwide is over 50% with it estimated to rise to over 70% in 2010 (Hanlon, 2008). In addition mobile phone hardware is increasing in computing power at a rapid pace (Koprowski, 2007). Unfortunately the mobile handset hasn't changed much from the traditional landline phone that still primarily has a numerical keypad and some direction keys. Camera based mobile phones see a penetration in the eighteen to thirty age bracket of over 96% globally (Wirefly, 2008). As the hardware and features deployed on mobile phones improves, so too must the interfaces which govern them (Kerr, Foster & Thinyane, 2008). This improvement in technological power has not been accompanied by related changes to the traditional 12 button interface. Some technologies such as trackballs, accelerometers and touchscreens have been introduced, but are limited to a small subset of mobile phones (JBenchmark, 2009).

Adapting the standard desktop mouse and keyboard style of interaction to a mobile device often encounters problems specifically in relation to the miniature size of mobile device input interfaces and screen size (Paelke, Reimann & Stichling, 2004). Adapting the analogue sticks on gamepads (found primarily on gaming consoles such as the Playstation and Xbox) to the mobile phone suffers from the same size problems. Improvements are being made in increasing the resolution on screens but the physical sizes of these screens are constrained by the size of the mobile device. Harper (2003) notes that increasing the size of the handset in order to deploy improved interaction techniques could make mobile handsets burdensome.

CHAPTER 1 – INTRODUCTION

Another problem arises when the lack of standardization is taken into account. Input methods, keypad layout and pointing devices all differ from one manufacturer to the next meaning a user is either stuck with one manufacturer or in the worst case has to learn a new interaction technique when moving to a new type of phone (Paelke, et al., 2004). Pointing devices on mobile devices are also of a low resolution (low pixel accuracy) when compared to the accuracy which can be obtained on the larger touchscreens, with a laser mouse and high resolution screen or with the analogue sticks available on major console gamepads such as Playstation 2 and 3 and Xbox 360. To address some of these problems related to mobile phone interaction a computer vision approach is proposed.

The small size of mobile phones means that the current major interaction technique to interact with them is the indirect key based input pad. The majority of mobile phones are equipped with the 12 key numerical keypad and directional keypad (plus other phone related keys). Phones such as the LG K360 feature a full qwerty keyboard as well as touch screen for numerical input and dialing. The Nokia 7380 features a scroll wheel interface similar to the Apple iPod. The user interacts with the phone by scrolling in a circular motion either clockwise or anti clockwise. This is similar to the original Blackberry phones which had scroll wheels similar to those found on desktop mice. Blackberry phones have always featured a qwerty keyboard and a track wheel or ball. These have given the Blackberry series pseudo mouse based interaction techniques. However these mouse based input techniques are limited to the Blackberry range of phones with the majority of phones using the simple numerical keypad input method. Some new phones such as the Samsung Omnia are adopting these types of interaction techniques but they are still limited to a small subset of handsets.

The need to keep handsets small but increase their usability has led to appropriately sized touchscreens being deployed on some handsets, most notably the Apple *iPhone* and Nokia products such as the *N81* (Lattimore, 2007). However, the main concern with these interfaces is that the user obscures the phone's screen with their fingers (Jenabi & Reiterer, 2008). This is a major problem when dealing with devices which are limited to a small screen size. As Watson (1993, pp 14) points out: "Gestures are [...] a natural and intuitive form of both interaction and communication" and hence should not impede on the very interaction they wish to enhance. Thus it is plausible that the convenience of mobile phones should be coupled with the convenience of gesture control to increase ease of use and the scope of applications. Various issues prevail with mobile interfaces and this is apparent in the multiple and ever changing interfaces which users are presented with. As Zerweck (1999, pp 25) states: "Gesture control has high potential to act as an input unit for predictable daily use in computer systems with spatial user interfaces".

CHAPTER 1 – INTRODUCTION

This need for more usable gestural and vision based interfaces has led to several recent studies that investigate appropriate vision based interaction techniques using the mobile phone platform (Bucolo & Billingham, 2007; Bhandari & Lim, 2008; Jenabi & Reiterer, 2008; Wang & Canny, 2006). However further studies are needed in this area especially in relation to finger based interactions. Finding a balance between speed and accuracy with the hardware available on mobile phones is of utmost importance for the user's experience to improve in analogue based interactions (interactions which allow variable input).

As noted touchscreens provide a solution but the main problem with this style of interaction is occlusion which is exhibited when the user's finger blocks much of the content on mobile device screen. Hence moving the touchscreen to the back of the handset would negate this problem (Baudisch & Chu, 2009). Unfortunately touchscreens have not been widely deployed on the back of mobile handsets with only prototypes showing the promise of this type of interaction (Baudisch & Chu, 2009). A back-of-device computer vision system would alleviate the problem of occlusion as well as the restrictions imposed on applications by the current keypad and pointing interfaces. The viability of this method of controlling mobile applications, as well as the benefits and possible problems are explored and an implementation to test this new technique of interacting with the phone is created. Certain difficulties encountered with current keypad interfaces such as lack of analogue input techniques could be alleviated by a user friendly finger-tracking interface. This would allow a broader scope of applications and games to be deployed on mobile phones.

1.2. Problem Statement

This study aims to investigate the viability of a vision based interaction technique using the camera situated on the back of the mobile handset. The technique will track the position of the user's finger to interact with basic phone functions and applications. This can be broken into four sub-problems:

1. Determine the current state of computer vision on both desktop and mobile phone systems in relation to deploying the finger interaction technique. This was conducted in order to determine the most suitable approach to deploy a mobile computer vision algorithm which is capable of finger tracking.
2. Design and implement an algorithm which will provide real time tracking via the camera on current mobile phone hardware and software API's in order to determine which software interfaces provide support for computer vision.

CHAPTER 1 – INTRODUCTION

3. Once a suitable platform has been determined it will be necessary to test the robustness of the finger interaction technique under different lighting conditions and background complexities in order to determine the most usable operational environment.
4. Investigate usability and user satisfaction with the finger interaction technique as compared to the standard mobile keypad

1.3. Scope of Research

A back of device camera enabled phone was required for this study. As mobile hardware differs greatly between manufacturers, a specific platform was required to allow a prototype to be developed. In this thesis the Symbian OS version 60 (S60) was chosen due to its high market penetration (Symbian, 2008). The N95 mobile phone was chosen because of its fast hardware. The N95 runs Symbian S60 with feature pack 1 installed. This thesis is therefore limited in scope to the S60 platform, although some software interfaces can be deployed on all systems, such as Java. S60 supports four major software APIs, these are: Java, Python S60, native C++ and Flash Lite. Flash Lite does not provide any interface to the camera stream and so will not be considered in this thesis. Many types of interactions could be implemented with the finger interaction technique, however this thesis will focus on simple positional input and not interactions such as text entry.

1.4. Research Methodology

Initially a literature study was conducted to explore the state of the art in mobile hardware and software, motion detection systems and multimodal interfaces related to finger interactions. These were critically analyzed to determine the best way in which to implement and deploy the system on a mobile phone. The advantages and disadvantages of each motion detection and optical flow algorithm were analyzed and a thresholding algorithm was chosen due to its ability to provide real time interactions with no pre processing (such as applying filters to a frame to find the edges) as this increases the time it takes to give feedback to the user.

The findings of the literature study led to the proposal of a suitable thresholding algorithm. Once this algorithm was created it was deployed to three programming APIs (Python, C++ and Java) and benchmarked to see which would provide the most suitable platform to deploy real time finger interactions on.

CHAPTER 1 – INTRODUCTION

A feasibility study was undertaken to identify the accuracy of the algorithm in various environments and at various light levels. These tests resulted in the identification of the best environment in which the system achieves the highest accuracy rating.

Having created a finger interaction system and tested its speed and accuracy, a user study was designed to test user satisfaction and various quantitative measures. Three applications were created, drawing on the literature study for the interaction style they should use. They were a target acquisition application to test mouse based interactions, a steering game to test analogue steering interactions and a directional gestures application to test gestural input. These applications were then used in the user study which collected both quantitative and qualitative data in the form of task times and error counts (during the study) and user questionnaires (after the study).

1.5. Summary of Findings

In the speed benchmarks C++ was found to be the only API capable of real time feedback. Bright fluorescent light and normal levels of sunlight were found to be the best lighting conditions with simple or textured backgrounds giving the highest accuracy scores in the accuracy tests. The user study explored mouse based interactions that would for instance be used in moving around a webpage in a browser or selecting an icon with a mouse. Steering based actions such as those that would be used in a racing game and directional gestures such as those that would be used to scroll through a menu or picture gallery. It was found through statistical tests that mouse style interactions presented very little difference between the finger interaction technique and the keypad. The steering based interactions achieved the best times and lowest error rates for the finger interaction technique. Users also preferred this technique over the keypad. The finger interaction technique performed poorly in the directional gestures tasks and the keypad was ranked to be better by the users. Although factors such as users prior experience with the keypad would have played a part in the better scores and comments it received. It was concluded then that steering based interactions were well suited to the finger interaction technique.

1.6. Thesis Organisation

This thesis is structured into several chapters.; The first chapter is intended as an introduction to the work, presenting background information and the problem to be solved. Chapter 2 presents the body of work relevant to this study. This chapter includes multimodal interfaces, computer vision technologies, optical flow and motion detection algorithms, advances in mobile technology and software interfaces available to mobile phones

CHAPTER 1 – INTRODUCTION

Chapter 3 introduces a solution to the current mobile phone interaction problem by using the camera to track the user's finger. The algorithm is introduced, explained and explored.

Chapter 4 explores the different API's available mobile devices, specifically on the Symbian operating system. A finger tracking algorithm is then implemented in these APIs. They are then benchmarked against each other in order to ascertain the relationship between deployability and speed.

Chapter 5 introduces, tests and explores the accuracy of the algorithm presented in Chapter 4. As lighting conditions can affect the accuracy of any vision based interaction technique, various conditions were identified and tested with the algorithm and problems relating to the light level and type of background are identified.

Chapter 6 formulates the results of a user study which was conducted to ascertain the overall effectiveness of the visual interaction system. Both quantitative and qualitative discussions are put forward and both the positive and negative feedback is explored and various statistics presented.

Chapter 7 provides a high level discussion of the problems addressed by the finger interaction technique and possible areas of deployment.

Chapter 8 concludes this document and provides insight into areas for future work.

Chapter 2 - Related Work

This chapter looks at the various fields related to computer vision and mobile device technologies. An initial overview is given of the section followed by an overview of multimodal interfaces and input. This will show the advantage of multimodal interfaces and how they can be applied in the mobile environment. This discussion is followed by an examination of current trends within desktop computer vision techniques and technologies, followed by an inspection of mobile device computer vision technologies. An explanation of various algorithms is presented in order to identify ways to deploy the finger interaction technique to mobile phones. Finally the available hardware and software interfaces are examined to ascertain the best way in which to deploy the new finger interaction technique.

2.1. Introduction

There is a trend towards new and innovative interfaces on mobile phones. This means that there is a large requirement for supportive underlying hardware (Kerr, et al., 2008). Faster processors allow faster image processing which in turn allows a myriad of other systems (such as motion detection) to be developed on the mobile platform. This increase in mobile technology has led to various successful studies in the field of motion detection and gesture control.

A recent study (Bucolo & Billinghamurst, 2007) uses a mobile phone's camera to control a handheld maze game. The phone is tilted to mimic the way in which a ball maze game is played. The camera interprets the angle at which the phone is being held at based on contrasting objects below the camera. There are however, very few games which use associated sensors to govern the user's interaction with them. Their study is one of the few which is beginning to look at associated sensors (sensors like cameras and accelerometers) which can be exploited on mobile phones. Various authors such as Wisniewski *et al.* (Wisniewski, et al., 2005) and articles such as Koprowski (2007) mention the need to better use the technology that is supported by current mobile phones.

One of the major requirements for gesture control and vision based interactions especially when working with the small screen of a mobile phone is that of preventing screen occlusion. Screen occlusion refers to the user blocking the screen with their fingers. This is particularly easy to do with the small size of the mobile phone's screen. Fortunately the majority of mobile phones have their cameras situated on the back of the unit. This is perfect for single handed gestures as the gestures will not obscure the screen.

CHAPTER 2 – RELATED WORK

The following section explores in more detail the various multimodal interfaces which have been deployed on mobile devices in order to identify the advantages a finger interaction system would provide.

2.2. Multimodal Interfaces

Multimodal interfaces process two or more combined user inputs, such as; gestures, speech, touch, head movements, key presses and so forth, in a coordinated manner with system output (Oviatt, 2002). The advantage of multimodal systems can best be described as the fact that: “multimodal interface design is inspired largely by the goal of supporting more transparent, flexible, efficient and powerfully expressive means of human-computer interaction” (Oviatt, 2002, pp 12). The ability to mimic the way in which people act and transfer data between each other every day leads to an intuitive interface. When people communicate with each other they make use of speech, gestures, facial expressions and body language in order to convey the information in the most informative way. While many of these types of inputs are not well suited to a mobile device it is important to note the power of being able to interact with an intelligent system using more than one type of input. Hauptmann notes that there is a large increase in efficiency when users use multimodal based input (speech and gestures) when compared with unimodal input (Hauptmann, 1989).

The standard keypad and direction keys present on the majority of mobile phones do not implement analog style interactions very well. This is because digital keypad buttons need to be pressed multiple times to for instance scroll a page or steer a car in a game. Hence a way in which one can classify a new interaction technique in relation to previous classification modalities is important. To this end various taxonomies and classifications are presented and their relation to the finger tracking technique discussed.

2.2.1. TYCOON

Martin (1998) presents a classification of modalities called TYCOON (TYpes and goals of COOperation). TYCOON is a theoretical framework for studying multimodality. The overall types of interaction and their goals can be summarized in Figure 2.1.

GOALS					
...					
translation	■				
recognition	■	■	■	■	
fast interaction	■	■	■	■	■
interpretation			■		■
learnability				■	
	transfer	equivalence	specialisation	redundancy	complementarity
	TYPES				

Figure 2.1: TYCOON classifies types of interaction along the bottom with the goals along the left side (Martin, 1998)

In Figure 2.1 the bottom row shows types of cooperation of modalities. The first column shows goals for improving human-computer interactions. As an example the first column shows that transfer between two or more types of modalities may be used to improve translation, recognition and interaction speed.

Translation is a goal which refers to the translation of information. For instance if two or more modalities cooperate by transfer a picture could be translated by one modality and passed to another for speech output. In this way a picture has been translated into speech. The recognition goal refers to the ability of the multimodal system to recognize input. For instance mouse click detection may be transferred to a speech modality in order to ease the recognition of predictable words (Martin, 1998). If the fast interaction goal is satisfied then the multimodal system is faster than using just one of the modalities. Interpretation refers to the user’s ability to understand the output from a computer. Learnability is the ease with which a user can learn the system.

In the following sections each type of cooperation presented in the above figure is explained and it is shown where the finger interaction technique fits into the classification of modalities. This will allow the identification of goals which will be satisfied by the finger interaction technique.

2.2.1.1. Transfer

When more than one modality interacts by transfer a portion of information generated by one modality is used by another. For instance a user will request information with one modality and receive an answer in another. This helps in the goal of translation. Transfer can also be used to improve recognition i.e. by using speech to help select an object with the mouse. Transfer can also help with fast interaction. One modality

CHAPTER 2 – RELATED WORK

can be edited by the user using another type of modality. For instance editing a spoken sentence with the keyboard. Transfer allows parallelism of processing in two modalities which allows for faster human-computer interaction

2.2.1.2. Equivalence

When several modalities cooperate by equivalence it means that each of the modalities is capable of processing the information as an alternative by either of them (equivalence can also be thought of as alternative). For instance the user can specify a command either by speech or using a mouse. Similarly in this study a user may use the keypad or the visual interaction technique to complete a task. Equivalence results in improved recognition. Equivalence also allows adaptation to the user by customisation allowing the user to change the modality to the one they prefer. Equivalence also allows faster interaction as the user can use the modality they are fastest with and are not constrained to a single form of modality which may not be suited to the task, such as the mobile phone keypad for steering based tasks.

2.2.1.3. Specialisation

Modalities cooperating by specialisation assign certain modalities to certain information which is always processed by that modality. Specialisation can help the user to interpret events produced by the computer as each modality can add semantic information and help the user's interpretation process. Specialisation can also improve recognition. Certain tasks which are limited to certain modalities can specialise more in their task as the search space is smaller and hence faster, resulting in faster processing (which leads to better recognition and faster interaction).

2.2.1.4. Redundancy

If modalities cooperate by redundancy the same information is processed by all the modalities. For example a user typing quit and uttering quit will negate the need for a confirmation box thus resulting in faster interaction. Redundancy also allows for a higher learnability in allowing the user to use two modalities at the same time to reinforce the action.

2.2.1.5. Complementarity

Complementarity of modalities means that various chunks of information are processed by each modality but must then be merged. Complementarity can enable faster interaction as the two interactions are processed at the same time. Improved interpretation is also a goal which can be accomplished by complementarity. This can for example be accomplished when a graphical output is presented for advanced users but is accompanied by a textual output for novice users.

2.2.1.6. Relation to Finger Interaction Technique

The proposed finger interaction technique cooperates through equivalence. This is because the finger interaction technique is proposed as an alternative to the keypad. The goals attained by this type of interaction are better recognition and faster interaction. The user would be able to use the keypad for tasks which are well suited to it and then switch to the finger interaction technique when analogue input is required.

2.2.2. Multimodal Gesture-Based Interfaces

In a study conducted by Pirhonen *et al.* (2002) on gestures used on mobile devices, it was found that gestures could be used to increase a user's performance (in this case, using the touch screen on a PDA) when interacting with the device. This was attributed to the user not needing to focus on the screen while using the keypad. The user can simply gesture (e.g. draw a line from left to right) to interact with the device and not have to concentrate on using a keypad.

Baudisch and Chu (2009) present a study into using the back of small devices as touch input. They state that current touch input on small screens suffers from the occlusion problem ('fat fingers problem') in that a large portion of the screen is blocked by the user's fingers (Figure 2.2 b). Moving the source of input to the back of the device frees up the screen for visual output (Figure 2.2 d). This is also true of the mobile phone and instead of waiting for touch enabled back-of-device inputs to emerge, the camera can be used instead and visual interactions developed to take advantage of the newly available hardware. As stated using the back of the device does not occlude the screen. This also allows more accuracy as the user can see exactly where their finger is in relation to the objects they are trying to interact with.

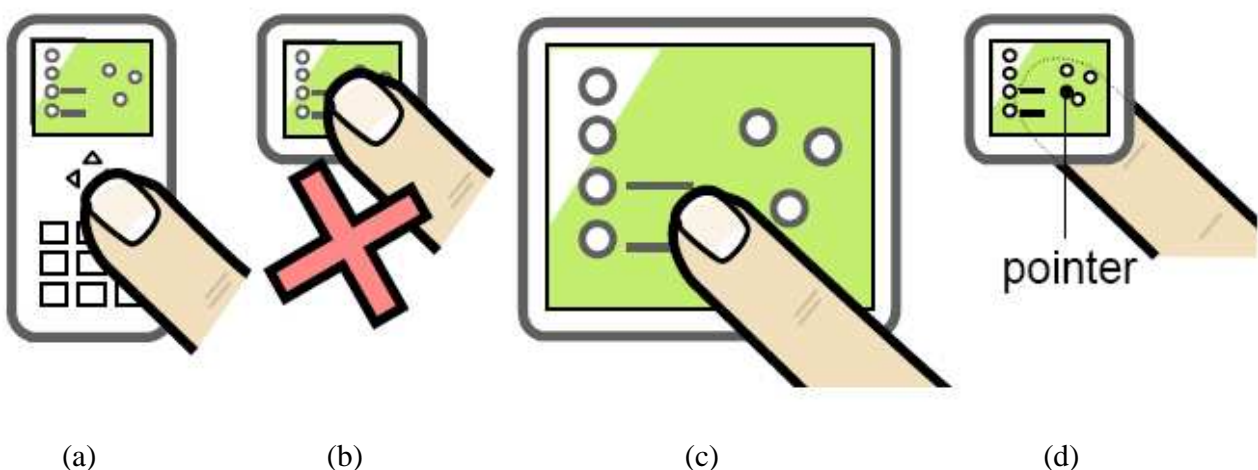


Figure 2.2: Different ways of pointing input are shown. (a),Using the d-pad ,(b)using a touchscreen on the same device (c) using a touchscreen on a larger device and (d) using the back-of-device touch input (Baudisch & Chu, 2009).

CHAPTER 2 – RELATED WORK

Back-of-device touchscreens have not yet been extensively deployed on mobile devices especially mobile phones. Cameras are widely deployed on mobile phones and hence provide an alternative for back-of-device interactions without the need for new hardware.

Lin, Goldman, Price, Sears and Jacko (2007) presented a study on nomadic data entry which is data entry while moving. Kjeldskov and Stage (2004) show techniques for evaluating mobile usability and comment on the nature of the mobile phone environment. Both studies note the difficulty of using a portable device in a mobile environment (lack of stable surface, visual attention being required elsewhere) but it is important to note a major advantage of the design of this study. The user is using a gesture based interface in a fluid environment i.e. the mobile device does not require a stable surface (even if one is available), as the user holds the device over their hand and controls it with gesture based input (similar to finger interactions). Lin *et al.* also note that performance on a mobile device is significantly reduced when walking through a complicated environment (e.g. down a street). In relation to this observation it is important to note that the amount of concentration required (especially in a mobile game) inhibits the user's ability to concentrate on outside stimuli. It is however, very unlikely that a user will engage in any complex actions with their environment when playing a highly complex game or application.

The concept of a multimodal interfaces should be at the fore when designing a new interface for complex interactions such as gaming. Gaming is one of the most interactively intensive processes both to a device and user, a highly flexible and usable interface must be created in order to create a seamless environment for interaction. It is with this in mind that different types of gestures are explored in order to better understand the background of gesture control and finger interactions.

2.2.3. Gestures Relating to Finger Interactions

While gesture recognition is not a new form of input for intelligent systems, it is still in very early stages of deployment on a large scale (Sharma et al., 1998; Bhandari & Lim, 2008). At their most basic level, gestures should be easy to understand by the user and the computer for them to be successfully used and interpreted by both (Long, et al., 1999). Various different types of gestures exist and are used by people every day. It is however, currently not possible for an intelligent system to identify and understand gestures as people do. It is important to be able to classify the various types of gestures which exist, McNeil, Levy and Pedelty (1992) classifies different types of gestures as follows:

- Iconic: Gestures which depict real objects and which are normally closely linked to the semantic content of speech

CHAPTER 2 – RELATED WORK

- Metaphoric: Similar to iconic gestures but referring to abstract ideas
- Deictic: Gestures which involve pointing to someone or something, whether real or imaginary.
- Beat: Binary gestures with only two phases(left/right, down/up)

These four types of gestures can be linked to speech in that they try to mimic meaning obtained from spoken interaction between two or more people. This is the closest and best way in which gestures can be used when interacting with a device and communicating intent. The other types of gestures are those which refer to and manipulate objects (Ergotic hand movements). These gestures are out of the scope of this study and will no longer be discussed. Sturman (1992) classifies hand gestures as the full use of all the capabilities of the hand for computer-mediated tasks which involve both position and shape of the hand. As Mulder (1996) points out many applications can be criticized for their choice of hand gestures but like most unresearched interfaces these choices seemed perfectly natural to the developer. It is with this in mind that this study aims to justify the choice of gesture used, i.e. single finger interaction. The initial choice of gesture can be said to fall within the category of beat based gestures. It is however different in that the gesture is designating more meaning than simply left or right as acceleration can also be taken into account. Mulder (1996) lays out seven distinct features which must be taken into account when designing software for gesture recognition. These are:

- Tracking Needs: The exact needs of the application such as the amount of accuracy required when measuring gestures.
- Occlusion: Fingers are relatively small and easy to lose within the background. The placement and orientation of sensors should take this into account.
- Size of Fingers: As stated previously fingers are relatively small and hence sensors should be small in turn and pick up a large amount of detail.
- Context Detection: The level of abstraction at which movement data should be interpreted i.e. speech/communication related or object related.
- Gesture Segmentation: When a gesture starts and ends. The maximum and minimum corresponding values based on digit or limb orientation must be identified.
- Feature Description: The features of the gesture which distinguish it from others.
- Gesture Identification: The underlying meaning of gestures which need to be interpreted.

CHAPTER 2 – RELATED WORK

These features are useful in classifying the types of gestures which should be used to interact with the mobile device (or any other device). The chosen gesture type for this study involves simple use of the index finger pointing away from the user. The user moves their finger and/or wrist in a sideways motion for steering based interactions and in all four directions for mouse based interaction (e.g. first person shooters). The need for gesture recognition is also simplified by the fact that a single finger is capable of far less articulation than an entire hand. If one sets out the seven features discussed above, in relation to the finger interaction technique it would be as follows:

- **Tracking Needs:** The application requires fairly accurate and fast variable based input on the horizontal plane.
- **Occlusion:** Phone sensors are assumed to be on the back of the unit and hence the screen is not obscured.
- **Size of Fingers:** Fingers are held close to the camera and hence negate their small size.
- **Context detection:** Detection should occur when and if a finger is detected pointing upwards in front of the camera.
- **Gesture Segmentation:** Possible calibration ranges should be instantiated. Otherwise the edges of the screen can be set as the maximum values.
- **Feature Description:** As stated the hand is orientated facing away from the user with the index finger or thumb pointing upwards in relation to the camera (which is held above the finger).
- **Gesture Identification:** Gestures are interpreted based on the application which is running i.e. a racing game will treat the finger as a steering wheel, whereas a page turning utility will turn a page when the finger passes a predefined maximum value (treat the gestures as beat gestures).

Reimann and Paelke (2006) note four important requirements when creating a gesture based interface for a mobile device. The first requirement states that the interface should be a fast and light weight system which would be best suited to the mobile environment. Secondly a system should include robust recognition as many mobile devices do not have high resolution cameras but this should not prevent gestures from being identified. Gestures should also be definable by the application and be easily identifiable by the system. It is also important to note (especially in the mobile environment) that users will not use gestures if they are embarrassing to use in public, i.e. require non discreet movements of limbs, head and body.

2.3. Computer Vision Based Technologies

This section presents and explores various fields within computer vision based technologies which can either be ported to or already run on mobile phones. Five main areas will be discussed, the first of which reviews the various desktop and PDA vision based technologies. Secondly, a brief review of the advances in mobile device technology is presented. Thirdly, image manipulation is explored, which deals with the editing of pictures at the pixel level on mobile devices which is important as it relates to the way in which camera data is interpreted by the mobile phone. Fourthly, a section on motion detection is presented which explores the use of the camera on the mobile device to implement computer vision. Finally, the use of mobile devices as mobile interfaces for other devices is explored, specifically when the camera is used to achieve this.

2.3.1. Desktop and PDA Technologies

Vision based interactions (including gesture control) have been implemented in various forms on the desktop platform, Bretzner, Laptev and Lindeberg (2002) explores hand gesture recognition. A multi-scale colour algorithm is used to detect blob and ridge features on a user's hand. Sections of colour are scanned for in the frame and these are collected and processed along with the orientation, position and scale of the hand in order to recognize the gesture being performed in front of the camera. Martin, Devin and Crowley (1998) presents active hand tracking at five frames per second. In order to detect skin the algorithm removes the luminance component of colours and creates a 2D histogram of the pixels from a region detected as containing skin. Freeman, Tanaka, Ohta and Kyuma (1996) discusses computer vision for interactive computer graphics. It is noted that computer vision algorithms must be fast, robust and must have the ability to be deployed on inexpensive equipment. To track large objects such as hands or fingers held close to the camera two frames were subtracted from each other to highlight the changes which had occurred in the latest frame. However this motion-based method requires the background to be stationary and hence is not suited to the portable nature of the mobile environment. Letessier & Bérard (2004) describes visual tracking of bare fingers for interactive surfaces, they note that using fingers allows more direct interaction with digital objects. They use a thresholding algorithm which then searches for the shape of the finger, which it identifies as a long rectangle. Tablet PC based gestures also show the potential of deploying gestures on the mobile platform (Microsoft, 2005), albeit the much smaller one of mobile phones. Because the desktop environment offers the most powerful hardware it is naturally the starting point for most new hardware intensive research. Hence, vision based principles formed on the desktop system have been ported over to mobile devices as seen in Arribas, Macia and Monasteria (2000) port of an

CHAPTER 2 – RELATED WORK

image manipulation system to a mobile device. Whilst the core computing principles (Processor, Memory etc.) stay the same between desktop and mobile device, the environment is different and hence the requirements for vision based interactions differ.

Various other motion detection systems have been developed on desktop systems. It is however prudent to note that many of these implementations use the same base assumptions and underlying workings as if they were deployed on mobile devices. For instance Bretzner *et al.* (2002) manually segment hands from the background using 2D histograms in order to identify skin colour in relation to background colours. Martin *et al.* (1998) show the need to combine various types of computer vision in order to create a robust and reliable computer vision system for real world use. Bretzner *et al.* note in their hand gesture recognition system that if certain postures of the hand are already known, then the shape of said postures can be easily identified and tracked in a complex environment by the intelligent system. As this study proposes more analogue interactions instead of gesture recognition there will not be a repository of known gestures to compare the camera frame against. However it will be assumed that the user's hand will be held with the finger pointing to the top of the phone parallel to the camera lens. This assumption means that the first section of skin detected is the tip of the finger.

PDA's have long been the closest mobile device to the desktop platform and are equipped with the ability to use stylus based gesture control (Pirhonen, et al., 2002; Lin, et al., 2007; Kjeldskov & Stage, 2004). The standard mobile phone with limited multimedia capabilities was combined with the more multimedia rich PDA. The resultant PDA hybrids were equipped with cameras (Figure. 2.3 b) and have seen significant growth in their hardware capabilities. Today the PDA is being replaced by the smart phone (Figure 2.3 c) which combines the processing power of PDAs with additional multimedia capabilities (i.e. digital camera style capabilities) of high-end mobile phones.

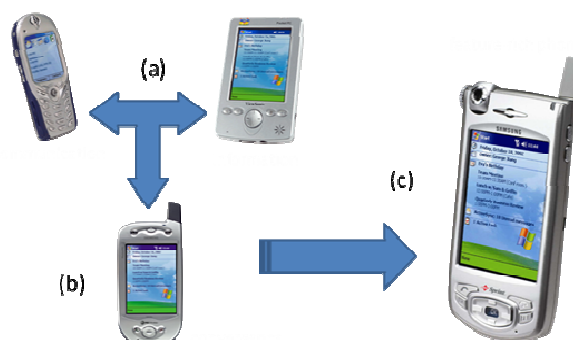


Figure. 2.3. Showing the combination of (a) mobile devices and personal digital assistants (PDA) called (b) XDA's. An amalgamation of all of these technologies is shown in (c) where both the task driven PDA and multimedia rich mobile phone are combined into one smart unit.

CHAPTER 2 – RELATED WORK

XDA's brought the processing power and large screen applications from PDA's to mobile phones. Smart phones have coupled the advantages of XDA's with both the high end multimedia functions, GPS and internet connectivity.

The following section explores image manipulation on mobile phones. This is important for camera based work on mobile phones as information from the camera is decoded into image data such as red, green, blue (RGB) values which contain values for each of the three colour components making a colour.

2.3.2. Image Manipulation

Various image manipulation libraries and utilities have been developed over the past few years and are now being deployed to mobile devices. These include technologies such as *Search by Camera! ER Search* (Research on Asia Group, 2006) which uses image recognition to compare images taken with mobile cameras and allow the user to obtain information on the subject of the photograph. A user takes a picture of a product (such as a wine label) and sends the image to a server which compares the image with pre-installed data and sends the user information of the product. Wilhelm, Takhteyev, Van House and Davis (2004) use a guessing algorithm (using a repository of annotations) to help with photo annotation on a mobile device which allows for intelligent tags and input to be generated. These examples aim toward a more flexible mobile environment for the user. This flexibility is coupled with the ability to interact in a much more unique fashion based on the hardware which is currently deployed on mainstream devices now or in the near future. For example, the Nokia N96's high quality camera optics will allow a high quality picture to be modified using the Nokia Computer Vision library (discussed in following section).

2.3.3. Motion Detection

Smart phones are deployed with the newest and fastest hardware available to mobile phones. Most notably accelerometers are now being deployed and smart phones such as the Nokia N95 and Apple *iPhone* now come equipped with this technology. Currently accelerometers are being used to orient the display according to the way in which the device is being held, but there are examples of accelerometers being used for other purposes, such as in mobile games (Wisniewski, et al., 2005).

Accelerometers are still limited to a fairly small niche of the mobile phone market. Hence the possibility of deploying these interaction techniques to a broader set of phones using camera based interactions is important. Accelerometers and touchscreens may still not be available to most phones but cameras are. It is with this in mind that various algorithms and camera based interaction techniques on mobile phones are presented in the hopes of identifying parts which could be used in the finger interaction system.

CHAPTER 2 – RELATED WORK

Beier, Stichling and Kleinjohann (2003) use an edge detection algorithm. In their solution 2D objects are extracted from the mobile camera feed using edge detection, using the Sobel Filter (a discrete differentiation operator, computing an approximation of the gradient of the image intensity function, see Figure 2.4). These 2D objects are then matched with known silhouettes of 3D objects. The authors use an approach first introduced by Dhome, Richetin, Lapreste and Rives (1989) which computes the relative orientation of the camera to the object using pose determination. As this is deployed on mobile devices the problem of computing the camera's orientation is simplified into lower dimensional sub problems. These are: Reduce the number of edges, assume limited camera orientation, classification of edges and using invariant relations. Similar objects are compared not only using edge detection but also texture mapping of areas of interest. Beier *et al.* (2003) conclude that their technique works on mobile devices and provides a reasonable (6-10 FPS) rate of marker-less and vision based Augmented Reality tracking. This system is an example of marker-less tracking.

Marker-less tracking is tracking without the use of a special marker such as that seen in Figure 2.5. Not using a marker requires other objects to be tracked in the camera view. These could be edges or objects. This is more difficult as the system may not have been exposed to such objects and so problems with accuracy can occur. Using a marker also known as a fiducial marker can lead to higher accuracy levels but does require the use of the stationary marker as a point of reference, this reduces the places in which the system can be used. This also requires a small setup time to place the marker.

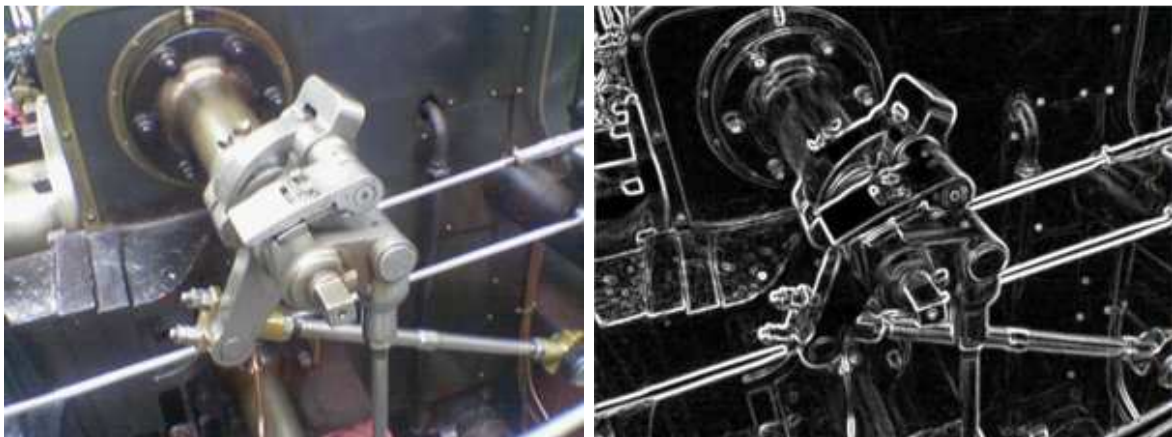


Figure 2.4: The original image is seen on the left. The same image which has had the Sobel Filter applied is displayed on the right, highlighting edges and aiding in edge detection.

This is not always ideal because of the nature of the mobile environment a user may be on the move and not have the time to setup a marker to interact with their device.



Figure 2.5: A marker based tracking system using a marker as a reference point for the camera

Winkler, Rangaswamy and Zhou (2005) identify the shortfalls of small form factor devices in relation to user interface design. They implement a camera based self-motion interface based on fiducial markers using the ARToolkit (Kato & Billinghurst, 1999) which allows the device to orient itself in 3D space. Winkler *et al.* (2005) implemented a racing game which uses the external markers to steer the vehicle within the game (Figure 2.5). They ran their application on a pocket PC with a 416 MHz processor and showed promising results in both performance and user acceptance. Rohs implements a similar marker-based system for mobile devices in (Rohs 2007).

Haro, Mori, Koichi *et al.* (2005) present new ways of interacting with mobile phones using the camera. Their motion detection algorithm is deployed on the Symbian OS and uses edge detection using the Sobel Filter (Figure 2.5). Haro *et al.* found that edges and corners are too computationally expensive to identify and hence corner-like features are identified and tracked. What Haro, Mori, Koichi *et al.*'s.(2005) algorithm allows is mouse like interactions using the mobile device. Examples include moving around a Quake III map using the camera as the mouse input, browsing through grids of pictures and playing 2D games with the camera acting as the directional keys. Further extensions by the authors can be seen in virtual environment interactions on mobile phones (Capin, Haro, Setlur & Wilkinson, 2006) and adaptive viewing on camera phones in (Haro, et al., 2005).



Figure. 2.6. Kick-up-menus are operated with the mobile devices camera and the users foot acting as a pointer.

A recent study by Bucolo and Billinghamurst (2007) uses a mobile phone's camera to control a handheld maze game. The phone is tilted to mimic the way in which a ball needs to be rolled in order to steer it around a maze. The camera interprets the angle by which the phone is being held based on contrasting objects below the camera. This is a computationally complex activity which is implemented without the use of accelerometers.

A mobile system called *Tinymotion* which is a product of a study done recently by Wang & Canny (2006) shows the potential of the mobile camera platform. *Tinymotion* utilizes the mobile camera to calculate various accelerometer-esque movements based on the movement of the phone. The camera data is calculated into various scrolling type commands on the phone. These commands allow the user to scroll through menus, text and play various games. This system is also capable of recognizing various gestures. In their user study positive feedback was gathered relating to using the mobile camera as an analog input device.

2.3.4. Taxonomy of Input Devices

Having established the coordination of modalities in relation to TYCOON in the previous section, this section presents a taxonomy of input devices based on the graphics based subtasks they can perform. Foley, van Dam, Feiner and Hughes (1990) and Foley, Wallace and Chan (1996) structure these subtasks as:

- Position: place an object in one or more dimensions
- Orient: move an object in one or more dimensions
- Select: Select an object
- Path: e.g. draw a line
- Quantify: specify a value

CHAPTER 2 – RELATED WORK

- Text Entry: enter text

Each of these tasks specifies the basic functions that a form of graphical interaction requires. This important taxonomy has been used and reused in various computer vision research projects to classify various forms of input computer vision techniques, whether in its original form or tweaked to fit the ever changing platforms computer vision interfaces are being deployed on. Orient and Quantify will not be explored in this study as the two dimensional nature of the finger interaction technique is not well suited to it. Text entry has been explored using the camera on mobile phones. An example of this is *tiltText* (Wang, et al., 2006) which uses the orientation of the camera and key presses to enter text. However text entry is out of the scope of this study.

Reimann and Paelke (2006) discuss various types of inside-out vision (The camera is manipulated in space and uses the video stream to control an application) based interaction techniques. Reimann and Paelke change Foley *et al's* (1996) taxonomy (Discussed in Section 2.3.4) to include gesture control and lay it out as follows:

- Selection: Selection from a set of options. When using inside-out vision the camera can be used as the selection tool where the object is selected by viewing it with the camera. Another example by Reimann and Paelke can be seen in (Paelke, et al., 2004) where kick-up-menus are used to select items (Figure 2.6).
- Position and Movement: The ability of a mobile device to locate itself in relation to six degrees of freedom (up/down, left/right, forward/backward, yaw, pitch and roll) or less. Examples include the Siemens *Mozzies* game on the SX1 smart phone (Reimann & Paelke, 2006). The *Mozzies* game overlays sprites (2D graphics) on a background provided by the mobile phone's camera and uses motion detection to point the crosshair at the 'Mozzies'(Figure 2.7 b).
- Quantification: Quantification of values as input parameters. E.g. using the camera as a mouse input for the mobile device. This task also includes augmented reality such as AR-Soccer (Figure 2.7 a) seen in (Geiger, Kleinjohann, Bernd & Stichling, 2001)
- Gesture: Gesture based interactions with mobile devices.



Figure. 2.7. (a) AR-soccer (Kick Real) is operated with the user’s foot as in the real game of soccer. Speed and direction are calculated when the user moves their foot to kick the virtual ball. (b): Mozzies game on the Siemens SX1 smart phone The user must move the phone and point the crosshair at the ‘Mozzies’ clicking on them to shoot them.

The following section explores mobile interfaces and their relation to computer vision on mobile phones. These interfaces use the camera as the form of input to control other devices. Various useful techniques can be drawn from this body of work in relation to creating a finger tracking system.

2.3.5. Mobile Interfaces

Mobile interfaces refer to the use of mobile devices which act as wireless interfaces or information gatherers for more cumbersome static devices e.g. servers and large public interfaces. These devices either are not equipped an interface or they are not very user friendly. Sharp (2004) details the use of mobile phones as portable interfaces for other devices. The technology uses visual tags (similar to fiducial markers) which are read by the camera and interpreted by the mobile device. The phone connects to the device using Bluetooth if additional interfacing is required. The visual tag (developed by High Energy Magic) is read in at 15 FPS and highlighted on the device’s screen using a crosshair. This is a good example of a multimodal interface (the camera acts as a mouse which is coupled with the phone’s buttons to act as mouse buttons) and displays one of the key future uses of mobile devices acting as mobile interfaces. A further example by Rohs *et al.* can be seen in (Rohs, Gfeller and Friedemann, 2006) where a similar camera based recognition system was created which analyzes visual codes with the camera which then identifies and processes the underlying meaning digitally in a similar way to barcodes.

Ballagas, Rohs and Sheridan (2005) implement two different types of interaction techniques for large public displays using the camera of a mobile phone. The first technique called sweep uses optical flow

CHAPTER 2 – RELATED WORK

algorithms to calculate the motion of the mobile phone and control a cursor on the public display, much like a mouse. A high latency (200ms) was experienced by the authors but as they point out the trend of increasing hardware speeds should negate this in the near future (The phone used in this study was a Nokia 6600). Also of interest is that the system uses the mobile phone's joystick to help the optical flow algorithm by initiating the direction in which the user wishes the cursor to move. The user then sweeps the phone and the cursor is updated via the mobile phones motion path. Because this system is made for public displays and not the mobile screen there is no problem with occlusion.

Jenabi & Reiterer (2008) detail a system for controlling large public displays using camera based finger interactions on a mobile phone. This system (called *Finteraction*) employs various finger gestures to manipulate output on a larger display (Figure 2.8). Finger gestures are used because of the screen absence problem (screen moves out of view when the phone is tilted to a certain degree) which occurs when using accelerometer style gestures. These gestures involve physically moving and tilting the camera which can lead to the screen not being fully visible at all times. For instance if the user tilts the mobile phone away from themselves then the screen will move out of view and the user cannot see it. The finger gestures proposed are:

- Moving finger left, right, up and down for cursor movements
- Tapping the lens of the phone for clicking actions
- Dwelling on an item to initiate a dragging operation
- Moving the finger closer or further away from the camera for zooming operations
- Moving the finger clockwise or anti-clockwise for scrolling operations



(a)

(b)

Figure 2.8: (a)An example of the proposed method of interaction employed by *Finteraction* using one hand as the method of input. (b) shows a closer view of the interaction technique

CHAPTER 2 – RELATED WORK

Ludwig and Reimann define the use of virtual objects within real scenes (Figure 2.9 c) as augmented reality (Ludwig & Reimann, 2005). The union of digital and real world information provides the user with important information when it is needed in both a clutter free and unique way. Further uses of augmented reality can include simulating planned objects by superimposing them where they would be built in the real world (Figure 2.9 b) and superimposing navigational data on a scene (Figure 2.9 a).

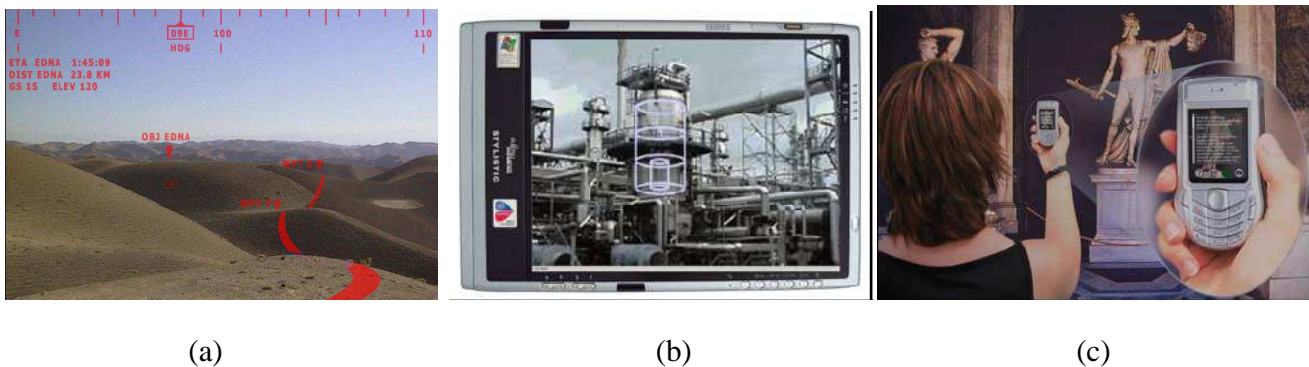


Figure 2.9: (a & b) Augmented reality allows virtual objects to be placed on top of real picture feeds to create a seamless environment for both simulation and completing real world objectives.(c) Virtual objects complement the real picture.(c) In this case information about the sculpture is presented to the user on their phone's screen.

Augmented reality systems are beginning to show how and where the virtual and real world can be merged in order to create a more diverse and easier to use environment.

The following section presents a set of motion detection algorithms which would be capable of running on mobile hardware is presented in the hopes of identifying parts which could be used in a finger interaction system.

2.4. Algorithms

This section details the major computer vision algorithms in relation to deploying the finger interaction technique on mobile phones.

2.4.1. Motion Detection

Drab and Artner (2005) discuss various types of motion detection algorithms when introducing motion detection on mobile devices. Motion detection algorithms identify motion within a series of images, normally from a video stream. These algorithms include:

- **Block Matching:** This algorithm divides an image into various equal sized blocks which are then used to identify the closest matching block to a reference frame i.e. where the block has moved to in the next frame.

CHAPTER 2 – RELATED WORK

- **Edge Detection and Tracking:** Edges are extracted from an image using either an Edge Detection matrix (Srihari, 1988), or the Hough Transformation (Shen & Castan, 1992). The detected images can then be monitored between frames and motion detected. A further example can be seen in (Beier, et al., 2003).
- **Analysis of Scene Components:** Objects are identified in an image and the following frames are searched for said object and motion is calculated (Koller, et al., 1994).
- **Projection Shift Analysis:** Frames are converted to greyscale and horizontal and vertical projection buffers are created for the luminance values of each row of pixels. Hence a change in motion will be reflected by the luminance buffers changing.

Drab and Artner (2005) offer evidence that their Projection Shift Analysis Algorithm is much faster than the other algorithms and hence is much better suited for the mobile platform.

2.4.2. Thresholding

Thresholding algorithms such as the one used in ARToolkitplus (Wagner & Schmalstieg, 2007) work with colours received from the camera. A threshold is defined and a reference colour is used to compare each coloured pixel received from the camera. If colours fall outside the threshold they are discarded, if they fall within the threshold then they have been detected by the algorithm. Thresholding algorithms are well suited to the mobile environment as they require no preprocessing and work directly on the stream received from the camera.

2.4.3. Optical Flow

The decisive paper on optical flow was published almost 30 years ago by Horn and Schunck (1981). This paper sparked a large amount of research into the subject of optical flow, such as how to benchmark optical flow (McCane, Novins, Crannitch & Galvin, 2001) which evaluates the performance of eight different optical flow algorithms. As shown in figure 2.10 the optical flow field is the field generated by the sphere which is turning from right to left.

Mitran defines optical flow problems as recovering the 2D flow from the intensity fields which are formed on the retina of a viewing device (Mitran, 1999). This problem can be seen as the inverse problem of estimating motion in an image sequence (without a tracking object) (Streeter, 2001). Barron, Fleet and Beauchemin (1994) define the problem as: “computing the approximation to the 2D motion field – a projection of the 3D velocities of surface points onto the imaging surface – from spatiotemporal patterns” (Barron, et al., 1994). Solving this problem is the core of motion detection and various algorithms exist to do this.

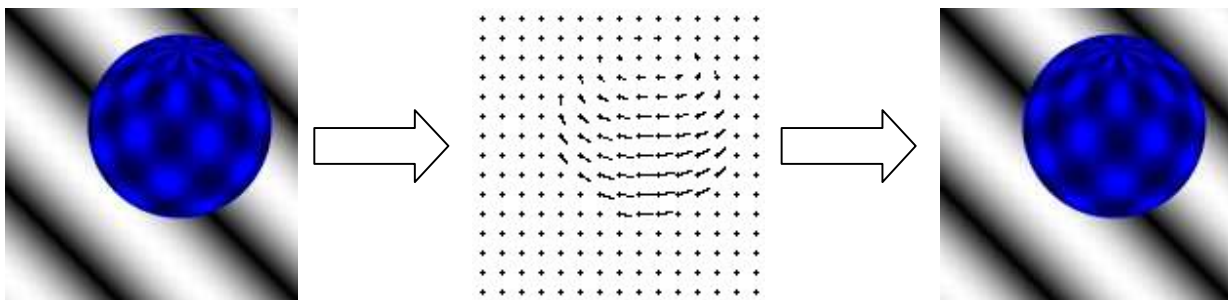


Figure 2.10: Optical Flow Field

There are numerous factors that can negatively affect the calculation of an accurate optical flow, such as: complex textured surfaces, camera noise, specular highlights (spotlight reflections on objects), change in lighting and transparency (McCane, et al., 2001).

The act of motion detection involves the projection of the 3D environment onto the retina of a viewing device such as a camera. This process is complex as tradeoffs must be made between accuracy and speed when trying to calculate 2D trajectories from 3D space and hence object motions (De Micheli, Torre & Uras, 1993). It is important to note that this study will not suffer from the majority of problems within the realm of the optical flow problem. This is because the gestures which the user uses scale well to two-dimensional space (particularly in the steering example). However, the hand is still a 3D object and hence must be processed from the 2D image produced by the camera from said 3D space. This in itself is no trivial matter but scales well to the 2D environment.

2.4.4. Types of algorithms

Barron *et al.* (1994) compare nine optical flow algorithms based on their ability to deal with density and accuracy. This is improved upon by Liu, Hong, Herman, Camus and Chellappa (1998) by comparing various other optical flow algorithms. It should be noted that there are many optical flow algorithms and they all trade off different aspects of motion estimation. Some provide good frame rates at high computational cost whereas others are fast but less accurate. Three algorithms were identified which could be modified to use in this study.

The variation of the Camus algorithm used in The NokiaCV library estimates optical flow from a series of bitmaps (NokiaResearchCenter, 2006) of low resolution; this, as discussed in the previous section is because there is a direct relationship between image size and processing time. For a device like a mobile phone which does not have the same processing power as a desktop it is very important that a good trade-off between accuracy and speed is attained.

CHAPTER 2 – RELATED WORK

Phase Correlation is used to measure the relative translative movements between two images. The algorithm works using a window function which creates a ‘view’ in the image and reduces edge effects. The movement between the two images is calculated using the Discrete Fourier Transformation (De Castro & Morandi, 1987).

The Horn-Schunck Method uses a global smoothness constraint to minimize the aperture problem (Horn, et al., 1981). The aperture problem exists when looking at a moving image through a constrained viewpoint such as a small circular aperture. In essence the problem manifests itself as the inability to identify the direction of movement correctly without viewing more of the picture. The Horn-Schnuck method boasts the advantage of generating a large amount of motion vectors, whereas the main disadvantage is that it does not deal very well with high noise levels.

The following section details the state of the art in mobile hardware technology. This is intended as an examination of the feasibility of deploying a computer vision solution for the finger interaction technique onto a mobile phone.

2.5. Advances in Mobile Technology

The previous section discussed the various algorithms which have been used for or ported to mobile phone computer vision solutions. This section examines whether current mobile phones have the technology to practically run the finger interaction technique.

Mobile phones have increased in core technology and branched into various specialized functions in the past few years. Early mobile phones simply had the ability to place calls and send short text messages (Guan, 2003). Displays were simple dot matrix screens, and these devices had slow processors and only a few kilobytes of memory (Figure 2.11 a). Screen display quality of mobile phones has progressively increased both in their colour depth and screen size (Figure 2.11 b), as has the memory capacity. Current phones have processors which reach speeds of 434MHz (e.g. ARM 11 in the Nokia N97) with Windows Mobile Standard/Professional Devices (small PCs some of which sport mobile phone type functions) having reached the 1GHz mark (GSMArena, 2009 b) and are equipped high resolution screens (Figure 3 c) which allow high quality camera applications to be operated such as image manipulation applications (Guan, 2003). There are now also gigabytes of space available to modern smart phones and up to 256 megabytes of RAM. These advancements allow vision based technologies to be developed and deployed on mobile devices which previously were not powerful enough or lacked embedded vision based technologies to support these interactions.

CHAPTER 2 – RELATED WORK

A trend towards new and innovative mobile interfaces means that there is a large requirement for supportive underlying hardware (fast processors and fast camera APIs) (Kerr, et al., 2008). Faster processors allow faster image processing (required for tracking with the camera) which in turn allows a myriad of other systems (such as motion detection) to be developed for the mobile platform. However, the tools with which these new systems are to be deployed must be able to access the new found capabilities of the underlying hardware. These advancements allow vision based technologies to be developed and deployed on mobile devices which previously were not powerful enough.

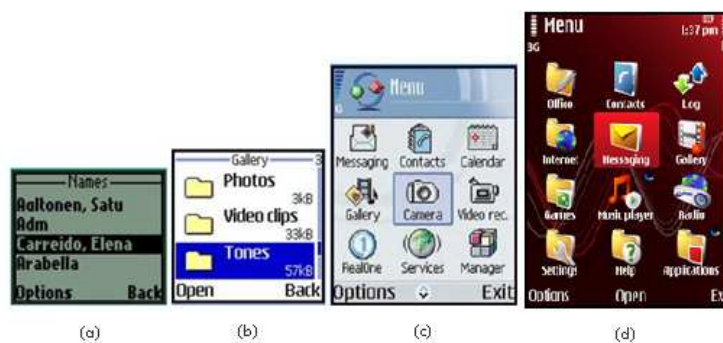


Figure. 2.11. An example of the progression of interface type and screen size in (a) Series 30 (84x48), (b) Series 40 (128x160) and (c) Series 60 (175x208) Nokia mobile phones. (d) Series 60 (240x320) latest Nokia smart phones like N96

2.6. Software Interfaces

This section turns away from hardware and focuses on the various platforms and software developments which are being deployed on current mobile devices. One of the most exciting for the fields mentioned in the previous section is the Nokia Computer Vision Library for C++.

2.6.1. NokiaCV

The Nokia Computer Vision Library is a library created by Nokia to facilitate the lack of support for camera and image manipulation in the Symbian Series 60 (S60) SDK (NokiaResearchCenter, 2006). The computer vision library provides key operations to process camera input, whether it is in video or static image form. This library provides a convenient test bed to develop camera based interactions which will work on Nokia phones, it is however limited to the Nokia phone line-up and hence is limited in its deployability.

The library provides various classes for manipulating images at the pixel level which include both math and matrix representations. The main class of interest is the optical flow class which can be used to detect motion in a scene. Direct access to the pixel stream is also useful as some APIs such as Java and Python do not provide this support. The move down to the pixel layer of image manipulation is very important as it

CHAPTER 2 – RELATED WORK

allows for any type of operations to occur on images and videos, which in turn leads to more flexibility and control over the image data. The speed of modern smart phones allows these manipulations to take place in the mobile environment, which because of processing power was not previously possible.

2.6.2. Java MIDP and the MMAPI

One of the major disadvantages of the NokiaCV library is that the source code is not available to the public and only works on Nokia phones. The Java platform however, only requires the JVM to run and hence can run on any phone which has it installed. This has led to Java being widely accepted on mobile devices as a development platform.

The latest version of the Mobile Information Device Profile (MIDP 2) introduces user interface and security enhancements, multimedia support, and RGB (Red Green Blue) image data manipulation (Guan, 2003). These enhancements allow developers to easily deploy interfaces, games and multimedia based applications to mobile devices. The MIDP runs on the Connected Limited Device Configuration (CLDC) which contains the virtual machine which the applications run on. Various optional API's can be run on top of the MIDP platform, such as the Mobile Media API (MMAPI).

The MMAPI enables MIDlets (deployed Java applications) to play and record audio and video data (Knudsen, 2008). This can be coupled with the MIDP 2's ability to manipulate images at the pixel level and apply transformations to work with a series of images for motion detection and gesture control. This is a far step from the initial Java deployments on mobile devices which supported very limited multimedia interactions and were really only suited to 2D games and basic applications. Java now supports 3D games as well as various API's which allow a developer to deploy multiple multimodal based applications to mobile devices.

2.6.3. Microsoft .NET Compact Framework

The .NET Compact Framework (CF) was released in 2003 and has become the leading platform for Pocket PCs and PDA hybrids, running Windows Compact Edition (Windows CE) such as the HTC and Smartphone (Wolfe, 2003). The CF is not very generic and this can be seen by the fact that the language runtime is embedded on devices in a memory foot print on a ROM chip. As with the NokiaCV library the applications created with the Compact Framework are limited and in this case will only deploy to devices which run Windows CE. The current release of the compact framework does not support pixel manipulation (Yakhnin, 2003). The next release of the Compact Framework (version 3) will include features which will better allow for motion detection and more usable interaction techniques, such as gesture control and pixel manipulation which are not catered for in the current release.

2.6.4. Python S60

Python is a well established scripting language on the PC and Mac. It has recently been ported to the S60 platform by Nokia adding various features specific to the mobile environment (camera, contacts, simple telephony etc.) (Nokia, 2009). To run Python a mobile phone must have the Python runtime installed on the system and scripts can simply be run from inside this runtime on any S60 device. However it is intended for fairly simple tasks providing the ease and lack of complexity of Python to the mobile developer. Chapter 4 - shows that a wrapper from the NokiaCV library to python allows for much more complex interactions to be performed, albeit at a slower pace.

2.7. Summary

This chapter has shown all of the recent developments on mobile devices which are related to vision based gesture interaction. The current strides in the underlying technology allow research into new human-computer interaction techniques, without additional upgrades in hardware to take place. Various optical flow and motion detection solutions were explored and analyzed in order to glean the necessary practical knowledge required for a finger interaction system. Creating multimodal interfaces with the use of gesture control would open many new possibilities for mobile applications. By carefully analyzing the progression of mobile device software (supported by the progression in underlying hardware) it can be better understood how to formulate a visual interaction technique which can take advantage of the widespread deployment of cameras on mobile devices.

Chapter 3 - Algorithm Overview

Chapter 2 presented relevant work to visual interactions on mobile phones. As shown in Chapter 2 there is a problem with pointing and steering style interactions on mobile phones, because the current keypad that is standard on most mobile phones was not designed for and does not work well with these tasks (Forman & Zahorjan, 1994). Because of the limitations of mobile phone hardware a careful balance between accuracy and speed must be established if a visual interaction technique is going to work practically. Hence the amount of additional processing tasks must be kept to a minimum; additionally the time taken to draw the results to the screen must also be suitably timely. Low speed (FPS) or low accuracy will result in an unsatisfactory interaction experience for properly interacting with applications. This chapter presents a thresholding algorithm which is used in this study to track fingers using the camera. The chapter contains an overview of the algorithm, followed by an explanation of each subsection (Camera, Input, Calibration, Tracking and Output), concluding with ways in which it can be tested and deployed.

3.1. Introduction

In Section 2.3 numerous computer vision projects and ways of capturing motion and tracking objects were presented. Throughout each study different interaction tools were used. This study uses a thresholding algorithm as seen in (Wagner & Schmalstieg, 2003) as well as pixel skipping and a search square (Song, Yu & Winkler, 2009). *Tinymotion* (Wang & Canny, 2006) for instance accomplishes many accelerometer-esque movements with the use of the back-of-device camera. These projects illustrated that computer vision interaction techniques could be feasibly and practically can be deployed on mobile phones.

Bare fingers were chosen as opposed to a stylus (Pirhonen, et al., 2002) or point of light as seen in (Mitran, 1999) because this presents the most natural means of tracking. Numerous other researchers have also chosen to use bare fingers (Hannuksela, Huttunen, Sangi and Heikkilä, 2006; Letessier & Bérard, 2004; Song, et al., 2009). Although supporting more natural interactions, bare finger techniques suffer from the following disadvantages compared to other tools. Stylus's provide better accuracy because they are much thinner than fingers and hence the tip is easier to detect when compared to the much wider tip of a finger. A light increases accuracy because it is very easy to detect as it stands out against the background. To make sure the algorithm is able to detect the finger it must first be calibrated to the users finger colour. This would not be required with a coloured stylus or light as the system would know the colour to detect beforehand. Figure 3.1 shows the proposed finger interaction technique which is similar to that proposed by

CHAPTER 3 – ALGORITHM OVERVIEW

Jenabi and Reiterer (2008) (see Section 2.3.5). The finger is positioned in front of the camera lens (Figure 3.1 a). The user can then see their finger on the viewfinder and use it to for instance move a pointer around the screen (Figure 3.1 b).

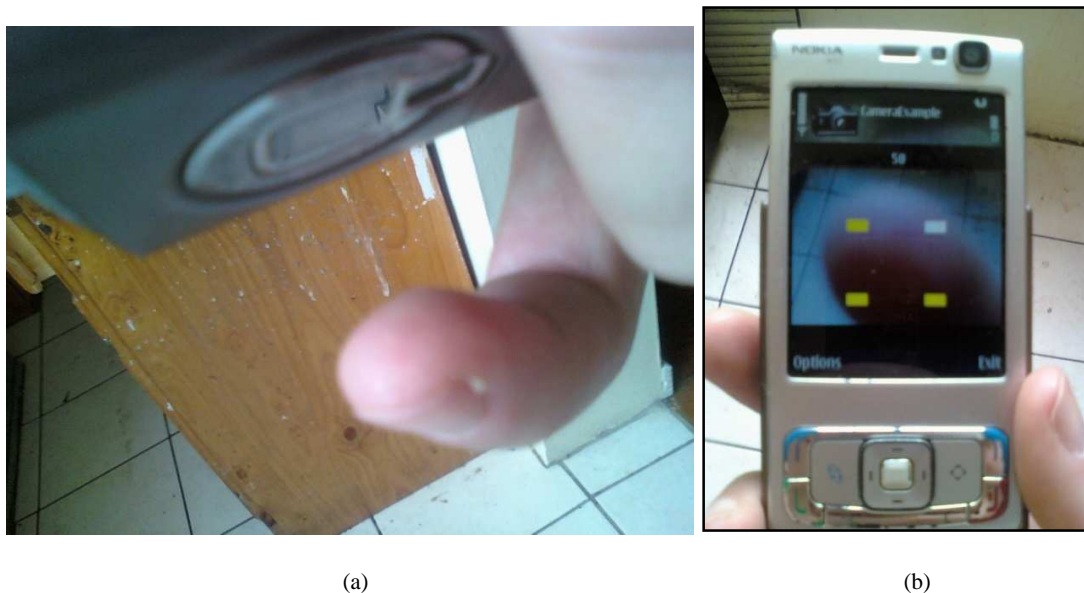


Figure 3.1: Part a on the left shows the single handed interaction technique from the side. (b) shows the technique from the top

After careful consideration, a thresholding algorithm was chosen because of its simplicity and proof of concept in ARToolkitplus (Wagner & Schmalstieg, 2003) and (Wagner & Schmalstieg, 2007). Thresholding algorithms work with grayscale images to segment pixels which fall above a certain threshold (background pixels) from those which fall within the threshold (object pixels) (Shapiro & Stockman, 2002). Figure 3.2 shows this segmentation with foreground pixels drawn in black and background pixels drawn in white. This helps distinguish certain objects in a scene from one another, in this case the trees, bridge and building are much easier to detect in 3.2 (b) than in 3.2 (a).



Figure 3.2: (a) shows the original image (b) shows the same figure with a threshold applied, highlighting foreground(object) and background pixels.

CHAPTER 3 – ALGORITHM OVERVIEW

Thresholding can also be used on colour images by designating a separate threshold for each of the RGB sections of an image, referred to as multiband thresholding. Pham and Schwock (2007) note that this reflects the way in which the camera sees colour but not the way in which humans do. Hence they propose using the CMYK (Cyan, Magenta, Yellow and Key/Black) colour model (used in printing) for image processing. For the purposes of this study using the thresholds on the RGB colours received from the camera is adequate as it alleviates the need to decode pixels to grayscale or CMYK.

To further decrease the load on the mobile phone hardware and increase the deployability, two augmentations were made to increase performance. These are:

- Pixel skipping and
- Search square

Both of these techniques were used to reduce the amount of pixels which had to be scanned within each frame received from the camera. Pixel skipping skips over pixels within a frame. So instead of scanning every single pixel one could for instance scan every second or fourth pixel effectively halving or quartering the amount of computations required in the scan of a frame.

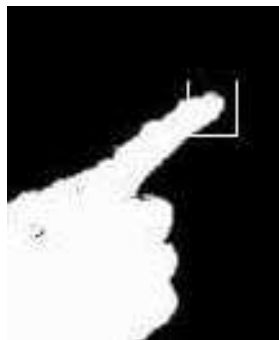


Figure 3.3: The search square is drawn around the finger. The next scan will only take place within this square (Song, et al., 2009)

The search square as seen in Song *et al.* (2009) places a square around the last detected position of the finger (see Figure 3.3). At reasonable frames per second (more than 6 FPS (Beier, et al., 2003)) the finger should not move a significant distance by the time the next frame has been acquired and is being scanned by the algorithm. Hence only the small square around the last known position need be scanned for the new position of the finger. This is discussed in more detail in Section 3.3.5. The following section details the multiband thresholding algorithm used for the proposed finger interaction technique.

3.2. Proposal of Algorithm

A multiband thresholding algorithm is proposed as seen in (Wagner & Schmalstieg, 2007) who use their ARtoolkitplus for mobile phones to perform marker based tracking using a thresholding algorithm. The multiband thresholding algorithm was chosen because it requires no pre processing and can work directly on the video stream received from the camera. Initially the user calibrates the algorithm to track the colour of their finger by holding it in front of the camera within a designated area. Once the user's finger colour is detected by the system tracking can commence. Tracking involves looking at each red, green and blue (RGB) pixel in the stream and compares each part of the colour to each part of the reference colour. Real-time feedback is required for this interaction technique.

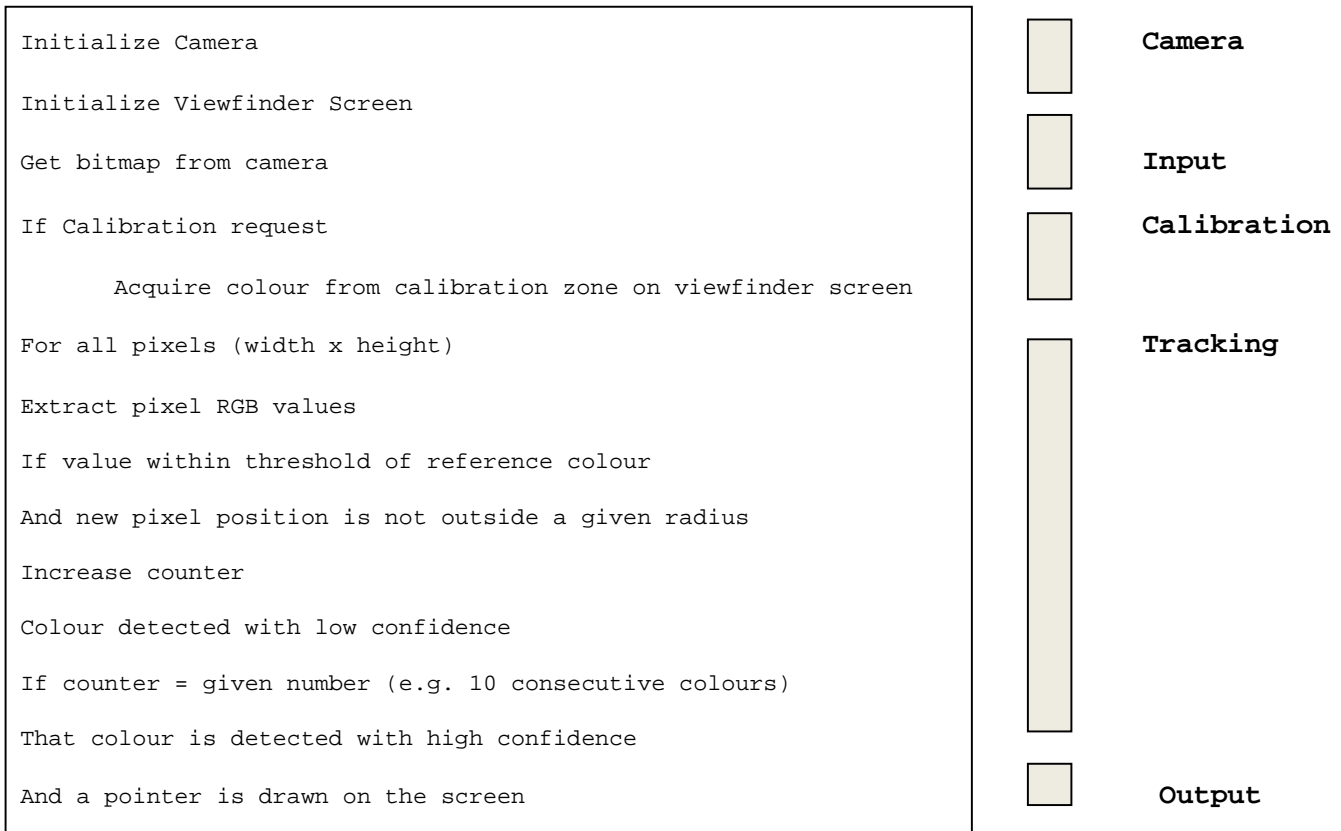
Pre-processing such as the Sobel Filter (discussed in Section 2.3.3) performs various actions on the image before it is scanned for movement. These pre-processing methods are used to aid in the accuracy calculations of the main algorithm by for instance identifying the edges in the image. Although the overhead associated with this step causes a decrease in performance. On desktop platforms this overhead is acceptable as there is so much surplus computing power. Because real-time feedback to the user is required, it was decided that pre-processing would hinder performance too much on a mobile phone, hence a real-time feedback algorithm was developed.

3.3. Overview of Algorithm

The entire process of tracking a finger can be broken into five distinct parts. These are:

- *Camera:* Camera support is provided by getting a stream from the camera (this same stream is used to draw the viewfinder picture)
- *Input:* Images are acquired from the camera and converted into a bitmap stream.
- *Calibration:* The user's finger colour is acquired for tracking
- *Tracking:* The user's finger is computed using a thresholding algorithm.
- *Output:* real-time output is presented to the user directly onto the viewfinder screen.

An overview of the multiband thresholding algorithm is presented in listing 3.1:



Listing 3.1: Pseudo code representing the multiband thresholding algorithm the bars along the right show the corresponding sub sections of the algorithm.

Each part of the algorithm is discussed in more detail in the following sections. This includes key sections of the code which are important to the overall solution.

3.3.1. Camera

The camera is setup through an interface and a viewfinder screen is displayed to the user. This viewfinder screen is simply a representation of the stream being received from the camera and hence it is easy to modify this stream to achieve the required output for real time interaction.

3.3.2. Input

Data from the camera is recorded and decoded into a bitmap stream. This stream comprises of RGB pixel data values which are not in positive Cartesian coordinates (x, y coordinates format) but a single stream of pixels 320 x 240 (76800 pixels) long. Because pixels are in a single stream and do not conform to normal coordinate systems used when drawing to the screen pixels directly above or below each other are 240 pixels apart which is important when trying to scan for pixels close to each other vertically. In the Cartesian

plane one pixel above (x, y) is (x,y+1) but with a stream the pixel above pixel z is the pixel number in the stream (z) + 240 (the width of the screen.).

3.3.3. Calibration

Calibration takes place when the user presses the OK button on the keypad. Once this request is sent, a square in the centre of the viewfinder screen appears. The user's finger should be positioned inside this square so that the average colour can be calculated. This is calculated by summing the red, green and blue parts of the RGB colour and dividing each by the amount of pixels scanned within the calibration square, this returns the average colour present within the square. Once this reference colour has been calculated the square disappears and the reference colour is passed to the tracking section of the algorithm.

3.3.4. Tracking

Tracking is the core of the algorithm and is responsible for finding the user's finger and then keeping track of it from one frame to the next. This involves scanning the stream of pixels and comparing the colour of the pixel to that of the reference colour. If a colour is detected which falls within the stipulated allowed range a counter is increased. Once this counter reaches a certain value (threshold) the finger colour is deemed to have been detected and output can commence. If the algorithm reaches the end of the stream and no colours are deemed to be the finger, then the last known position of the finger is set as the position of the finger. The finger is always detected when the user calibrates the system as the colour being calibrated is obviously present. So if no finger colour is ever detected the pointer will stay in the centre of the screen. If the system loses track of the finger colour but had detected the finger in the previous frame, then the position of the pointer in the previous frame will be drawn on the current frame.

Each pixel in the stream is compared to the reference colour. Table 3.1 shows the reference colour and two possible stream colours which would be acquired while scanning through a single frame. Stream colour one is received first and stream colour two second. Assuming that the stream is read in and stream colour one in Table 3.1 is the first pixel in the stream then the computation to ascertain if stream colour one is within the reference colour threshold will be $x - y > z$ where x is the red section of the reference colour, y is the red section of stream colour one and z is the threshold value. If $x - y$ is greater than z then the colour falls outside the threshold and is discarded. So if $z = 20$ then using the values in table 3.1 results in $55 - 10$ which is greater than 20 and so this colour will be discarded. The next colour in the stream, stream colour two is passed to the algorithm and the following series of comparisons are made:

- The red part of the RGB colour is compared:

CHAPTER 3 – ALGORITHM OVERVIEW

- 55-69 is not greater than 20
- The algorithm compares the green section
- 23 – 29 is not greater than 20 so finally the blue part is compared, which results in
- 19 – 24 which is also not greater than 20 and so this colour is tagged as a finger colour.

Table 3.1: Stream colour one and stream colour two represent two possible colours which the algorithm may receive. The red, green and blue components are shown below the actual visual colour. The reference colour is shown in the same manner for comparison.


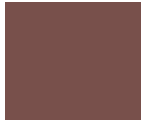

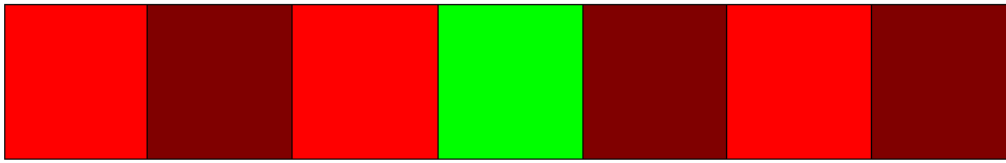
Stream Colour One			Stream Colour Two			Reference Colour		
								
Red	Green	Blue	Red	Green	Blue	Red	Green	Blue
10	23	19	69	29	24	55	23	19

Figure 3.4 illustrates the way in which the algorithm uses blocks of continuous colour to identify the reference colour with a high confidence. This makes sure that the colour is a substantially sized block (in this case 7 pixels or wider) of colour and not just a speck (one to six pixels) which the algorithm may pick up as a false positive. In part (a) two sections of red are effectively split by a coloured pixel which is of a high contrast to the other red pixels (in this case a single green pixel). If the amount of consecutive pixels of similar colour is seven then the two sets of three red pixels will be passed over in the algorithm until a section of red as seen in (b) is found. In this case the final pixel in (b) will be the centre of the pointer.



(a)



(b)

Figure 3.4: Two samples of seven continuous pixels from the stream. (a) Shows three pixels of red within a given threshold on either side of a green pixel. A counter will reach three on the third pixel but will be reset to zero when the fourth pixel fails to fall within the threshold. (b) Shows seven colours all falling within a given threshold. Once the counter reaches seven the colour is detected with high confidence.

As can be seen in Listing 3.2 the size of `smallComp` dictates the accuracy of tracking the reference colour. A higher value translates to lower accuracy in that colours received from the stream are more likely to return a false value as they fall outside of the specified threshold. The core of the algorithm is the `if` statement. This statement's conditional contains the code which subtracts the current stream colour from the reference colour for each of the RGB components. The number obtained from this operation is then compared to `smallComp` to ascertain whether or not that colour is within the threshold.

If all three components of the RGB value return true (are within the threshold) then the counter is increased by 1 and a second comparison is made between this counter and `constTest`. `constTest` specifies the number of consecutive pixels required to result in a high confidence detection. This means that the location of the pixel in the stream is used by the output part of the algorithm to draw the pointer.

If one of the colour components falls outside the threshold (is greater than `smallComp`) the conditional will return false and the counter will be reset to 0. This means that 7 (the value assigned to `constTest`) consecutive pixels must be found again in order for a high confidence detection to take place.

```
//if greater than this then not close to colour
Assign smallcomp = 20
Assign constest = 7

Get RGB pixel data
Get last known position
Initiate search square with centre at last known finger position
If the current pixel is greater than smallComp then not the finger colour
    Break and get the next pixel
    If the current pixel is less than smallComp then the colour has been found with low
confidence, add 1 to the counter
    if counter equals constest
        colour found with high confidence
    Draw the pointer
otherwise
    Current section of pixels has been broken, reset the counter
```

Listing 3.2: Pseudocode depicting the initial assignment of colours followed by the core tracking algorithm

3.3.5. Search Square

In order to maintain a high frame rate, a search square was implemented to decrease the amount of operations required to scan for a finger. A search square is defined as a square with its centre positioned at the last known position of the pointer. The algorithm will first scan inside this square for the finger colour before scanning the whole stream. Once the user's finger is detected it is unlikely that it will move a substantial distance in front of the camera in the time it takes for the next frame to be scanned by the algorithm. With this in mind, a search square was implemented to reduce the amount of pixels scanned in each frame and hence increase the performance of the system. This method is the same as that used by Song et al. (Song, et al., 2009) in a vision-based 3D finger interaction system for games. It is used to reduce the amount of processing required to locate the finger in each frame. Figure 3.5 shows a 64 pixel stream with a screen width of 8 pixels (the phone uses a 76800 pixel stream with a screen width of 240). The colour being detected is red, the amount of consecutive pixels required before detection is set to two and the finger has been detected at pixel 29 and a pointer drawn in yellow. The yellow border shows the current block (top left at pixel 2 and bottom right at pixel 56) in which the next scan will take place. As soon as the pointer at 29 was drawn the next frame would have been requested but instead of starting the scan for the

CHAPTER 3 – ALGORITHM OVERVIEW

finger at pixel 1, the scan would start at pixel 10. If the colour is not found by pixel 56 then the scan is started from 1. Hence the four divorced pixels of red in the top left corner will not be detected unless nothing is detected within the scan block.

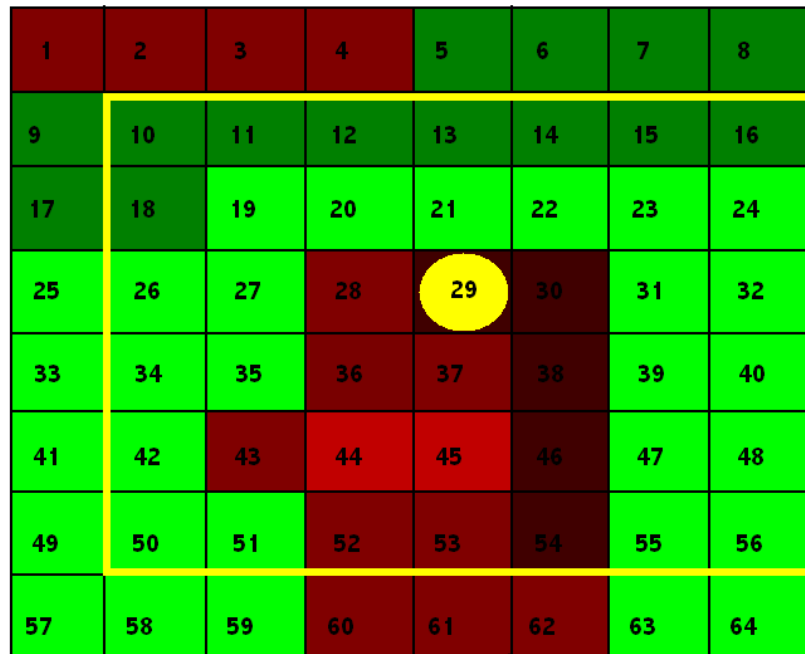


Figure 3.5: a 64 pixel stream showing a detected finger at 29 and the yellow border showing the search square for the next frame

This method of scanning reduces the total amount of pixels scanned drastically, which in turn increases performance. However if the user's finger is not detected within the block then the scan must start from pixel one adding additional computation. This is a troublesome situation and a high framerate will be required to negate this possibility.

3.3.6. Output

In the output phase, the position of the pointer is displayed on the viewfinder screen where the user's finger is detected. This is a simple output algorithm which draws the pixels back to the backbuffer (the region in memory where the next frame is being stored, drawing to this effectively draws to the next frame). As stated in Section 3.3.1 the stream is modified before being displayed to the user.

3.4. Summary

This chapter has presented an overview of the pixel thresholding algorithm used to track user's fingers on a mobile phone. The algorithm reads in a bitmap stream from the camera and access the RGB values to perform simple thresholding calculations. This algorithm works within a lab setting but this must be proved and tested to work within real world situations. Various different programming environments are supported by the Symbian S60 operating system and provide different ways with which to access the camera and generate feedback to the user. It is not clear how these programming environments will perform with the thresholding algorithm. To this end, the next chapter presents a series of experiments which implement and test this algorithm in three of the most prominent programming languages supported by the S60 platform.

Chapter 4 - Performance Analysis

This chapter presents a series of benchmarks for the three most prevalent programming languages on the Symbian operating system (C++, Python and Java) (Nokia, 2009) to see if they are capable of accessing and using the hardware available on mobile phones for intensive vision interaction tasks (such as motion detection and image processing). In addition, attention is also paid to the potential scale of deployability to other mobile operating systems and hardware.

4.1. Introduction

The desktop environment has the most powerful hardware and so it is naturally the starting point for most new hardware intensive research. Hence numerous vision based principles formed on the desktop system have been ported over to mobile devices (Arribas, et al., 2000). Because of the processing limitations on mobile phones one approach is to offload the main processing to a server as seen in AR-PDA (Geiger, et al., 2001). A client (mobile device) offloads the majority of the processing over a network to a static server with more processing power. The only work done by the mobile phone is capturing the image and rendering the output. Once computation on the server is completed the data is then sent back to the mobile device. While this solves the major processing speed problem it raises additional problems such as the need for a continuous broadband connection to the server. The other major problem is the latency which occurs over a network which is not suitable for precise pointing tasks and interactions required for games and applications. Hence this type of processing approach will not be explored as a solution which runs on the device is required for widespread use. The following section details the various benchmarks used to test the performance of the thresholding algorithm on the three programming APIs.

Regardless of the platform, benchmarking is used to allow a comparison to be made across different interaction techniques. Within the context of image processing on the mobile phone platform, the following benchmarks have been used.

Kaxiras, Narlikar, Berenbaum and Hu (2001) conducted a study into mobile phone processing speed, hardware and 3D graphics. In addition, Akenine-Moller and Ström (2003) benchmarked the power consumption and types of processors on mobile phones. These studies show that mobile phone environments are vastly different to desktop systems as processing power is much less powerful than desktop systems. Other factors such as powerful graphics and physics accelerators are not possible to implement on mobile phones because of size and heat issues. Graphics accelerator chips are starting to be

CHAPTER 4 – PERFORMANCE ANALYSIS

deployed on mobile phones but are still orders of magnitude slower than those found on desktop systems. Hence benchmarks on the mobile phone platform are important to gauge the effectiveness of mobile hardware and software. Akenine-Moller and Ström (2003) stress the need for more advanced architectures and hardware to better render graphics on mobile phones.

Wagner and Schmalstieg (2007) perform various marker based tracking benchmarks using their ARToolkitPlus for mobile phones. These benchmarks also used a thresholding algorithm for tracking and were conducted on various high-end mobile devices and on the desktop platform.

Spmark released by Futuremark allows phones to be benchmarked in the areas of 3D game performance, 2D game performance, image processing, battery life and various tasks which are commonly used in Java applications. (Futuremark, 2009). Similarly, JBenchmark is a utility used to benchmark the Java ME capabilities on mobile phones and includes benchmarks for scalable vector graphics, CPU tests (which include business math, chess, game physics, image processing, shortest route search, XML parsing, ZIP compression), 2D graphics, 3D graphics and composite performance (JBenchmark, 2009).

4.2. Methodology

In this study a custom set of benchmarks were designed and created to ascertain the speed with which each of the programming APIs could perform visual based tasks. A custom benchmark was used because no benchmark exist which will run on all three APIs and measure the image manipulation based tasks that need to be tested. Hence benchmarks were created drawing on the currently available benchmarking systems mentioned in the previous section. All variables outside of the different APIs were kept constant with the various tasks that were completed to allow a correlation to be made between speed, accuracy and the API. All benchmarks were recorded using internal timers with a resolution of 1 micro second. Results were then converted into seconds and frames per seconds for the various comparative charts. The five benchmarks tested the speed of acquiring the image from the camera, the speed with which the application could scan through the image, the latency and the time to track an object. The phone was also ‘warmed up’ as initial frames received from the camera could take up to three times longer than normal to acquire as it has to be initiated for image capture. The benchmarks used are laid out below:

- **Frames Per Second (FPS):** The time taken to retrieve a 320×240 bitmap frame from the camera and then save it, making it available for the motion detection algorithm as in (Schmalstieg & Wagner, 2007).

CHAPTER 4 – PERFORMANCE ANALYSIS

- **Latency:** A video set to run at exactly 1 frame per second was viewed in a cradle by the phone's camera. The phone was positioned in the cradle so the lens was 15cm away from the monitor. The phone was parallel to the surface of the screen and was pointed at the centre of the screen. The video is of a high contrast red dot on a white background moving from one location to another on a horizontal axis. The algorithm detects the dot in its initial position. The dot will move one frame later on the horizontal axis to the right by 100 pixels. This can be detected on the mobile phone by counting the pixels moved from the left edge of the screen by the dot. It is then possible to subtract one second from the time obtained when the dot is detected in its second position to obtain the latency as in (Sielhorst, et al., 2007) and (Kerr, Thinyane & Foster, 2009).
- **Full Scan:** The time taken (in seconds) for a frame to be loaded, every pixel scanned (320×240) and an assignment of each pixel's RGB red, green and blue values to the same variable. This is to test both reading and writing of pixel data to memory. The input to the camera did not matter in this test but it was placed face down to create a consistently black background. The test was then executed using internal timers. Values were converted from microseconds to seconds for comparison sake. Similar to the image and memory processing tests in JBenchmark (JBenchmark, 2009).
- **Pixel Skip 10, 30, 50:** This benchmark involves testing the APIs under the same conditions as full scan except that the specified amount of pixels are skipped by the algorithm, reducing the amount of pixels scanned. This is a process used to reduce the overhead when performing motion detection. Every x^{th} pixel is skipped and so less computational steps are needed to scan the entire image as in (Schmalstieg & Wagner, 2007).
- **Time to Track:** Time to track a red dot on a white background situated in the center of the viewfinder. Red was chosen as a high contrast colour to the white background. The device was cradled in front of the image on the screen (as in the latency benchmark). Every pixel was scanned starting at the top left and moving down to the bottom right. The delta time (time taken to display one frame measured in seconds) is subtracted from the final time to give the raw time to track. This benchmark is highly dependent on the system's ability to read and write pixels from a bitmap stored in memory as in (Schmalstieg & Wagner, 2007).

From the above benchmarks there are two other values which can be calculated:

CHAPTER 4 – PERFORMANCE ANALYSIS

- **Dropped Frames:** Records the number of frames dropped due to latency (caused by the motion detection algorithm) and is measured in frames per second. This is calculated by multiplying the latency by the FPS.
- **Delta Time:** The time to display one frame, calculated from FPS.

For comparative reasons in Section 4.3 some values are converted from seconds to FPS. FPS can be calculated by $1 / \text{delta time}$ so the resultant FPS can be calculated by $1 / (\text{delta Time} + \text{benchmark time})$ so if the delta time is 0.05 seconds and the time to track or full scan time is 0.05 seconds then FPS can be calculated as $1 / (0.05 + 0.05) = 10 \text{ FPS}$.

4.2.1. Platform

This section describes the platform on which the various benchmarks were conducted as well as providing a brief introduction to the APIs used.

The benchmarking experiments were conducted on the Nokia N95 smart phone. The N95 has a Dual ARM 11 332Mhz processor including a 3D Graphics HW Accelerator and 64MB RAM. The N95 is equipped with a built in 5 Mega Pixel camera which uses a Carl Zeiss autofocus lens, which is considered a high quality camera for a mobile phone (GSMarena, 2006 a). The phone was updated to the latest version of Symbian OS supported (i.e. S60 4rd Edition Feature Pack 1). The N95 provided a platform which was ranked as the fastest phone in the JBenchmark image processing tests and fastest in terms of overall processing power at the time of the study (JBenchmark, 2009).

4.2.2. APIs

The three languages tested were C++, Python for S60 and Java 2 Micro Edition. These are the three primary languages available on the Symbian Series 60 OS (along with Flash Lite 3.0 which did not support access to the camera at the time of the study) (Nokia, 2009). Each language was tested using the same multiband thresholding algorithm discussed in Section 3.2. This algorithm was used to obtain pixels from a bitmap image supplied by the mobile phone's camera and then detect a block of colour in the image, which has been calibrated by the user.

The latest stable version (at the time of the study) of each API was used, these are:

- Java 2 ME environment, MIDP 2.0 CLDC 1.1
- Python (PyS60) 1.4.5
- C++ for Symbian S60 3rd edition feature pack 1 SDK.

CHAPTER 4 – PERFORMANCE ANALYSIS

Each of these languages supported different ways of obtaining frames from the camera. A snapshot (the same function used to take photos) function is used for the Java implementation as this is the only supported way to obtain frames from the camera. In Python a newly available wrapper to NokiaCV (Nokia Computer Vision) was used to obtain the frames from the camera (NokiaResearchCenter, 2006). The wrapper gives access to the stream from the camera. This is done with `sPycNokiaCVImage` which creates an image which can be converted into a `cfbsbitmap` recognized by python. This allows frames to be extracted from the stream. Once the frames are obtained each of them is scanned with standard python functions and `viewfinder.waitForViewfinderFrame()` is called to get the next viewfinder frame. Listing 4.1 shows the initial process of converting a viewfinder frame into a workable bitmap and then extracting three RGB pixels.

```
sourceBmpImage = viewfinder.backBuffer._bitmapapi()
    sNokiaCVImage = sPycNokiaCVImage.NewLC()
sPycNokiaCVImage.Bitmap_CreateL(sourceBmpImage)
    (c1,c2,c3) = sPycNokiaCVImage.GetPixel(i, j)
```

Listing 4.1: The initial process of obtaining a bitmap in Python using the wrapper to the NokiaCV. The first line grabs a frame from the viewfinder. The second line creates a new instance for the bitmap to be saved. The third line creates a bitmap out of the backbuffer frame and the third line assigns the red, green and blue sections of pixel (i,j) to c1, c2 and c4 respectively

C++ provides a set of functions which fetch frames as bitmaps and saves them for processing. A frame is received by the algorithm as a pointer to the bitmap `CFbsBitmap* aBitmap`. One can then assign the pixels by using the `TRgb` variable with the required pixel and then assigning each separate section of the RGB colour by for instance calling `RefRedData=pixel1.Blue()`. This pixel is an integer from 0 to 255 and can now be used by the algorithm for various threshold comparisons.

4.2.3. Algorithm

The motion detection algorithm used in the benchmarks is the thresholding algorithm discussed in Chapter 3 -, which compares RGB values within a given threshold colour. Thresholding in itself is not new on the mobile platform and is used in ARToolkitPlus for marker based tracking (Wagner & Schmalstieg, 2007). A high level representation of the pixel comparison algorithm used in some of the benchmarks is the same as that laid out in Section 3.3.4.

4.3. Results and Discussion

This section presents the results recorded from the benchmarks. All results are summarized in Table 4.1 and it can be seen that C++ has the highest FPS followed by Python (10.53) while Java exhibited a very low FPS (0.2). Delta time (time to display one frame) is low for Python and C++ and high for Java at almost 6 seconds. C++ achieved the fastest time in Time to track at 0.05. Java achieved 3.93 seconds and Python achieves 9.92 seconds, the slowest time. Full scan, which can also be thought of as pixel skip 1 as every pixel is scanned showed the same pattern as time to track. C++ was fastest (0.02), Java was second (0.29) and Python the slowest (20.22). Pixel skip 10, 30 and 50 showed the same pattern with speed increasing as the distance between scanned pixels increased. Latency (the time for a movement in front of the camera to be registered on the viewfinder) for C++, Java and Python was 0.05, 0.5 and 2.25 respectively (lower is better). Dropped frames (amount of frames not shown in a second) for C++, Java and Python was 0.95, 0.08 and 23.66 respectively.

Table 4.1: Benchmark results for Python, C++ and Java. Values are specified as either FPS or seconds with standard error in brackets.

Benchmarks	Python	C++	Java
FPS (FPS)	10.53(± 0.53)	18.20(± 0.91)	0.20(± 0.01)
Delta time (seconds)	0.09(± 0.01)	0.05(± 0.001)	5.97(± 0.3)
Time to Track (seconds)	9.92(± 0.5)	0.05(± 0.002)	3.93(± 0.2)
Full Scan (seconds)	20.22(± 1.01)	0.02(± 0.001)	0.29(± 0.01)
Pixel Skip 10 (seconds)	1.14(± 0.06)	0.01(± 0.001)	0.17(± 0.01)
Pixel Skip 30 (seconds)	0.345(± 0.02)	0.001(± 0.0001)	0.147(± 0.01)
Pixel Skip 50 (seconds)	0.2156(± 0.01)	0.0004(± 0.000001)	0.1428(± 0.01)
Latency (seconds)	2.25(± 0.11)	0.05(± 0.001)	0.50(± 0.03)
Dropped Frames (FPS)	23.66(± 1.18)	0.95(± 0.05)	0.08(± 0.001)

The next section presents a detailed discussion of the results for each of the tests and draws conclusions based on the software interfaces and programming APIs available on the current mobile platform.

4.3.1. FPS

A series of t-tests shows that there was a significant difference between all the FPS values for each API (C++ and Java ($t = 77.45$, $p < 0.001$), Java and Python ($t = 40.38$, $p < 0.001$) and C++ and Python ($t = 19.88$, $p < 0.001$)). Table 4.1 shows that C++ is the fastest (18.2 FPS) followed by Python (10.5) and then Java (0.2 FPS). The reason that Java scores so low in the FPS benchmark (0.2 FPS) is because there was no proper function to directly access the frames received from the camera. A snapshot had to be taken at 640×480 pixels using the camera shutter (using the `snapshot ()` function) and hence takes up much more time than the methods used in C++ and Python. As expected, these values are still not as fast of that of a PC which can run at over 30 FPS scanning every pixel (Mitran, 1999).

4.3.2. Time to Track

The time to track benchmark's results are most pertinent and practical in respect to actual motion detection in a mobile environment. As stated in Section 4.2, the time to track is the time in which it takes the algorithm to detect a dot in the middle of the camera's field of view. This benchmark is highly dependent on the ability of the API to read and write from memory. The benchmark also involves a six part binary 'AND' comparison for the purpose of detecting the dot. A t-test was run on each of the APIs and a significant difference was found between all values (C++ and Java ($t = 18.46$, $p < 0.001$), Java and Python ($t = 19.01$, $p < 0.001$) and C++ and Python ($t = 61.86$, $p < 0.001$)).

In Table 4.1 C++ performs well but does slow down by 8 FPS from the initial FPS (shown in Figure 4.1). The FPS in Table 4.1 for receiving data from the camera is 18.2 hence the 8 FPS drop when tracking in the time to track benchmark. Java shows a large slowdown needing almost 4 seconds on average to scan half of the bitmap in order to detect the dot. The FPS drops from 0.2 to 0.1 ($1 / (5.97 + 3.93)$) However, this is 6 seconds faster than Python S60 which took 9.9 seconds in this test and dropped from 10.5 FPS to 0.1 ($1 / (0.09 + 9.92)$) FPS a significant drop. It can be seen that even though Python S60 is able to maintain a respectable (more than 6 FPS (Beier, et al., 2003)) FPS it slows down significantly (9.9 seconds) when performing this benchmark (Figure 4.2). From these results it can be seen that the functions used by Python are not well suited to reading and writing bitmap data at speed and performing fairly large binary operations.

4.3.3. Full Scan

Figure 4.1 shows the results of the full scan which scans every pixel in the bitmap and rewrites each pixel back into the stream at the same memory address. This means that the APIs need to be able to quickly read and write data but not conduct any binary comparisons as with the time to track benchmark. Mobile phones

CHAPTER 4 – PERFORMANCE ANALYSIS

are known to have slow memory read/write speeds (Schmalstieg & Wagner, 2007) and so this benchmark tests memory operations to quantify the read/write speed for each API. The figure shows frames per second and full scan for each of the APIs. This allows the difference in mean performance to be clearly identified. A paired t-test was performed to determine if there were any significant differences between the FPS when under no load and the FPS when running a full scan. Only Python exhibited a significant decrease in performance ($t = 38.586$, $p = 0.2 \times 10^{-10}$, $df = 9$). C++ and Java did not exhibit a significant difference in speeds.

It can be seen in Table 4.1 that the difference between time to track and full scan shows that Python is being affected more by multiple memory operations than binary operations. Full scan is 10.3 seconds slower for Python which equates to a 67% decrease in performance when moving from a time to track test to the full scan test. Both C++ and Java show an increase in performance when compared to the time to track test. C++ is 0.03 seconds faster in the full scan test which equates to an increase of 28%. Java is 3.64 seconds faster in the full scan test which equates to an increase in performance of 86%. So Python is being hindered more by memory read/write operations than by binary operations. C++ and Java exhibit the opposite of binary operations decreasing performance (in Java's case by a large margin of 86%). It is prudent to note that this test shows that both memory operations and binary comparisons slow down the system. Only in the case of C++ is there any possibility of real time feedback to the user as Java and Python are too slow.

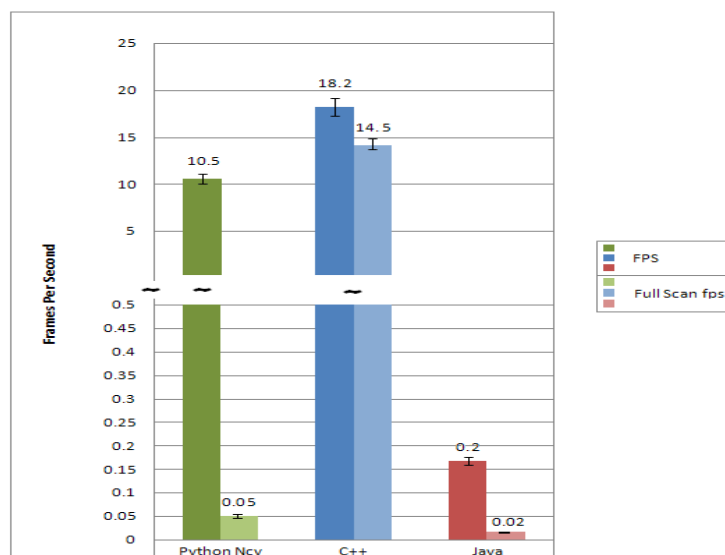


Figure 4.1: Frames per second shown for each language, encompassing values whilst under load (waiting for the image, saving it and scanning the whole bitmap: Full scan) shown in darker shade and under no load (simply waiting for the image from the camera and saving it) shown in lighter shade.

4.3.4. Pixel Skip

In this benchmark tests were run with pixel skips of 10, 30 and 50. As can be seen in Figure 4.2 there is a consistent decrease in mean performance. A one-way ANOVA was run on each of the APIs to identify where any significant differences in the data occur. A significant difference was detected within the Python data, $F(4797.035, 3879.12) = 1.2366$, $p = 0.00$. Tukey post-hoc comparisons of the Python dataset were conducted to detect what the significant difference was. There was a significant decrease in performance from a full scan (pixel skip 1) to pixel skip 10 ($MS = 0.25475$, $p = 0.000159$) and again from pixel skip 10 to pixel skip 30 ($MS = 0.24375$, $p = 0.002613$). There was however no significant difference from pixel skip 30 to pixel skip 50 ($p = 0.959$) for Python. Python's performance increases dramatically from a full scan to scanning every 10th pixel. A significant difference was detected in the C++ data. A Tukey post-hoc comparison showed that a significant increase in performance occurs when moving from pixel skip 10 to pixel skip 30 ($MS = 0.00001$, $p = 0.00159$) or pixel skip 50 ($MS = 0.00001$, $p = 0.000159$) but there is no difference when moving from skipping 30 pixels to skipping 50 pixels or from a full scan to skipping 10 pixels. The only significant increase in performance for Java was moving from a full scan to skipping 30 pixels ($MS = 0.01216$, $p = 0.017373$) or 50 pixels ($MS = 0.01216$, $p = 0.036283$).

Figure 4.2 shows the time in seconds of each scan taking 20.2 seconds to perform a full scan and 1.1 seconds to scan every 10th pixel, this translates into a jump from 0.05 FPS to 0.9 FPS. Every 30th pixels take 0.3 seconds which translates to 3.3 FPS. It is prudent to note that this is still not a manageable frame rate. Even when only scanning every 50th pixel the frame rate increases to 4.64 FPS. At this speed and accuracy only simple, general motion detection tasks (scene movement detection) can be performed.

Java performs fast at full scan speeds and sees a comparatively large jump from full scan to scanning every 10th pixel jumping from 3.45 to 5.82 FPS. Increasing to scan every 30th pixel results in the frame rate increasing to 6.85 FPS. This is a decent speed for general motion detection (Wang, Zhai & Canny, 2006) and the accuracy is not so low as to negate practical use. It is however not accurate enough for finely detailed tracking such as tracking a finger or stylus. If the initial camera fetch functions can be sped up and properly implemented to obtain frames directly from the camera then this would be a very good result in terms of deployability (as Java is available and supported on many more phones than Python and C++) and speed for general motion detection.

C++ benefits from comparatively large gains when moving from a full scan to skipping pixels (Full Scan takes 0.015 seconds to complete whereas Pixel Skip 50 only takes 0.0004), but the raw gain in frame rate is

CHAPTER 4 – PERFORMANCE ANALYSIS

rather small compared to the other API’s as it was fairly fast in the first place. C++, like Java, is limited by (bottlenecked) the rate at which the hardware can deliver the images from the camera.

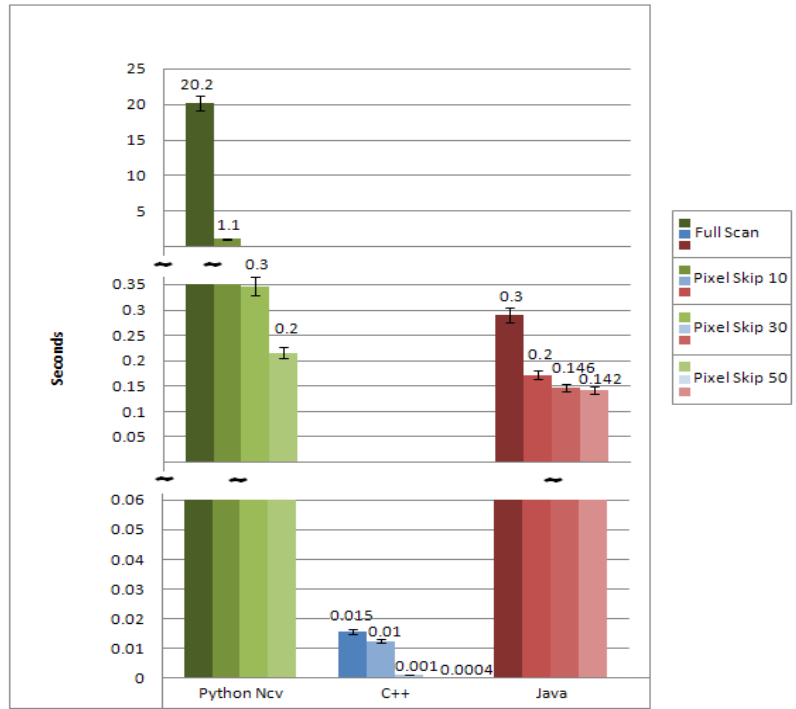


Figure 4.2: Results for the various pixel skipping benchmarks. Full scan equates to every pixel being scanned. Pixel Skip 10 equates to every tenth pixel being scanned and so forth. Values are ranked darkest to lightest from Full Scan to Pixel Skip 50.

4.3.5. Latency

Latency is directly linked to user’s performance in real time interactive systems such as augmented reality systems (Lai & Duh, 2004). It has been proven that there are significant reductions in speed and accuracy in systems which exhibit a high latency (Sielhorst, et al., 2007). Hence the values obtained for the Latency benchmark are important for gauging the overall performance and hence practical usability of the API.

Table 4.1 displays the overall latency of all three APIs. Again, the same ranking as in the other benchmarks are observed, with Python having a high latency of 2.246 seconds. Java can also be seen to have a fairly high latency of 0.5 seconds. C++ is ten times faster than Java and exhibits a latency of 0.052 seconds, thus the C++ implementation falls within acceptable latency speeds for visual interaction systems (Sielhorst, et al., 2007).

4.3.6. Speed and Deployability

An API’s range of deployable systems, operating systems and phones can be thought of as its deployability or breadth. The speed at which the API processes the various benchmarks can be thought of as the length.

CHAPTER 4 – PERFORMANCE ANALYSIS

Figure 4.3 shows the length and breadth for all the APIs tested. This figure shows that C++ has the lowest deployability but the highest speed. Python shows moderate deployability and a low speed. Java displays a large deployability and moderate speed (not counting initial FPS).

C++ can only be deployed on a phone which carries the same version of Symbian (i.e. Symbian S60 3rd edition and the same feature pack) as was used when coding the program. This limits the deployability of the application to a very specific platform.

Applications created in Python require an up to date version of Python S60 installed on the mobile phone. Although this does not limit the application down to the version and feature pack of the Symbian OS, it does however limit the application to run only on phones running Symbian S60 (many Nokia phones)

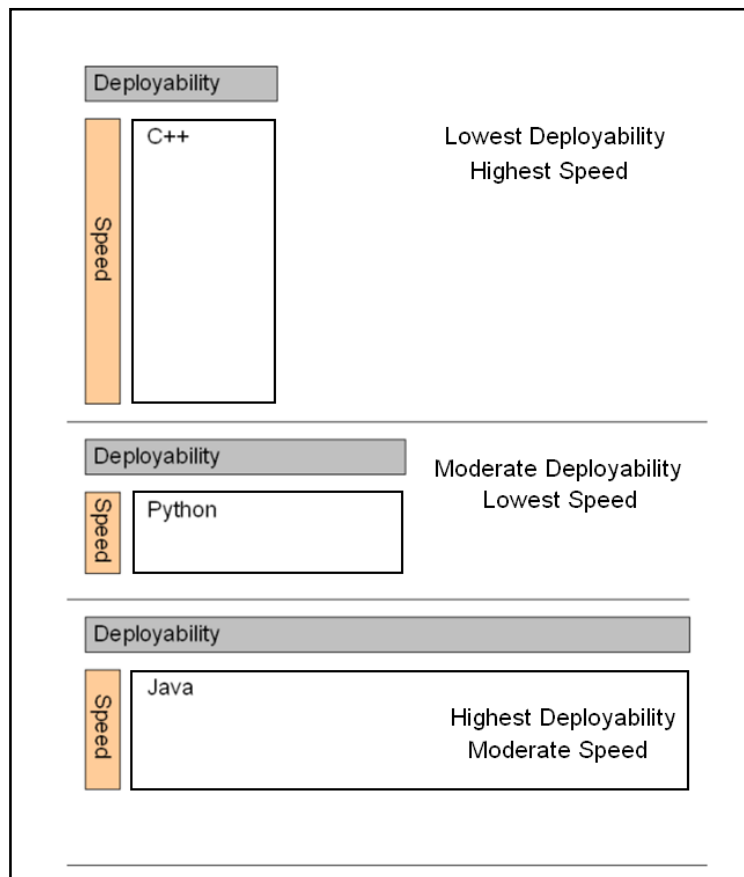


Figure 4.3: Each API's deployability (breadth) and speed (length) on the mobile environment.

Java requires a device to have the Java run time environment installed. In the case of the benchmark it would also require a phone which has the advanced multimedia API (MMAPI available on all new phones (Guan, 2003)). This is a limiting factor in deployability but is still not as restrictive as C++. This allows applications created in Java to run on a much wider mobile device platform.

CHAPTER 4 – PERFORMANCE ANALYSIS

Java would be a very promising choice for intensive bitmap interactions if it was not for the limited way in which camera interactions are handled. Although, Java shows some fairly impressive results especially when compared to Python, it is let down by the length of time it takes to receive a frame (snapshot) from the camera.

4.4. Summary

This chapter has shown the various performance differences between the three popular mobile APIs on the mobile platform. A series of five benchmarks were performed using the functions which are available and the results tabulated and charted. From these results it was possible to glean speed and practical deployment statistics.

It was found that coding at the lowest level in C++ produces the fastest results. It was also noted that coding at this level drastically limits the deployability of the application. C++ was also the only API to provide a truly practical real time experience with regard to motion detection and intensive pixel level bitmap operations.

Python, implementing a wrapper to the Nokia CV class was able to obtain the initial frames from the camera at a respectable speed of just over 10 FPS. Unfortunately the functions available to the Python programmer do not seem to be able to read and write bitmap data at a reasonable usable speed as was seen by the overall speed drop to 0.05 FPS when scanning the entire bitmap.

Perhaps the most important discovery was that of Java's performance with regard to reading and writing bitmap data. It was only the speed at which Java obtained the initial frames that severely limited the practicality of this API.

Evidence has been presented that mobile hardware is now at the stage of supporting vision based interactions to be deployed. Unfortunately memory write speeds and GPU acceleration still needs to increase further as these are responsible for many of the processing bottlenecks(Schmalstieg & Wagner, 2007).

Across all of the benchmarks the C++ API was proven to be the only environment in which finger tracking could be implemented with real time feedback. While speed is an important part of computer vision it is useless if the system cannot accurately detect the user's finger from one frame to the next. The following chapter examines the system's accuracy through a series of experiments.

Chapter 5 - Accuracy, Colour and Lighting

The nature of mobile devices dictates that they will be exposed to many different environments when being used. These environments sport different lighting conditions which can range from direct sunlight to complete darkness (Paelke, et al., 2004). As mentioned earlier (Section 2.2.2), all vision based interaction techniques are susceptible to interference in the visual channel, including such issues as occlusion, the primary visual interference that this study is concerned with is lighting conditions and background complexity. Objects which are tracked by a camera are susceptible to many problems brought about by varying lighting conditions. Objects may change in appearance simply because of small changes in light level and shadows moving in and out of view (Moeslund & Granum, 2001). The effect of light on the tracked object also applies to all other objects in the frame of view. Changes in background or light or a combination of both can affect the accuracy of a computer vision system. It is with these differing lighting conditions and background types in mind that an accuracy study was conducted. The study was run to ascertain the accuracy of the finger tracking algorithm laid out in Chapter 3 and further tested in Chapter 4. To test the algorithm's accuracy a set of videos were captured at different light levels recorded. The light levels were determined in an initial light level study which was conducted to ascertain the average lighting level in environments sporting fluorescent and tungsten lighting. These videos were used to benchmark the algorithm in terms of accuracy. These results were used in the user study (Chapter 6) to provide users with an environment which allows the finger tracking system the highest accuracy. Negating any errors due to the system.

5.1. Introduction

Moeslund and Granum (2001) characterize three types of motion capture systems (surveillance, control and analysis) and the three main performance parameters required by each (robustness, accuracy and speed), as seen in Table 5.1. Speed was examined in Chapter 4, this chapter will examine the accuracy and robustness of the system. The finger tracking system falls into the control category (Table 5.1) as it is used to control mobile phone functions. Surveillance and analysis systems will not be explored in this document.

The robustness of a system is said to be its sensitivity to changes. A control system may require a fairly high level of robustness depending on the assumptions of the environment it is to operate in. A control system operating only indoors under bright lighting would require a fairly low level of robustness, whereas the finger tracking system requires a fairly high level of robustness because of its mobile nature and the

CHAPTER 5 - ACCURACY, COLOUR AND LIGHTING

amount of different lighting conditions and environments it will be subjected to, hence robustness can be assumed to be 1 in Table 5.1.

The accuracy of a system is said to be how close the captured motion is to the motion performed by the user. Accuracy is also said to be directly proportional to the size of the object being tracked (Moeslund & Granum, 2001). Table 5.1 shows that speed (discussed in the previous chapter) is the most important performance parameter in a control system, followed by robustness and accuracy.

Table 5.1:The three application areas and their requirements of the three main performance parameters. Robustness in a control system is tied to whether or not lighting and background are within a controlled environment or not. If it is within a controlled environment then robustness is not important, if it isn't then robustness is an important attribute of the control system as it must adapt to the changing environment it is used in.

	Surveillance	Control	Analysis
Robustness	2	1/0	0
Accuracy	0	1	2
Speed	1	2	0

The accuracy study used different types of backgrounds in the sample videos. Both Moeslund and Granum (2001), Lenman, Bretzner and Thuresson (2002) identify uniform and complex backgrounds within computer vision tasks. Uniform backgrounds presenting the easiest tracking environment and complex backgrounds the hardest. Hannuksela *et al.* (2006) record sets of videos on both textured and complex backgrounds when testing finger tracking. Hence for this study the three different types of backgrounds chosen were complex, textured and simple(uniform). These backgrounds were kept constant with only the light level changing. Three of the most common lighting conditions were used (The Engineering Toolbox, 2005). These were direct sunlight, tungsten (incandescent) and fluorescent lighting. Different levels of sunlight were observed for all backgrounds namely dark, normal and bright, these light levels were matched to previously recorded sunlight levels which are standardized (ISO) exposure values (EV) which can be converted to lux values (which measures luminance, the intensity of light perceived by the human eye) for comparison (The Engineering Toolbox, 2005).

5.2. Methodology

This section will explain in more depth the backgrounds used, the way in which they were recorded and why they were chosen. The five different lighting conditions under which the system was likely to be used are also discussed. Once this study was completed a series of special cases were run in order to test the algorithm in environments which were perceived to cause accuracy problems. To find the light values used in the main experiment an initial light level study was conducted.

All EV values were taken with an ambient light meter (Minolta Auto Meter III F) at ISO 100 which allows conversion to other units such as lux and foot candles. This light meter uses a hemispherical sensor which gives readings for scene luminance. Once the correct lighting was observed the scene was filmed with the N95 mobile phone and the individual frames were later passed to the algorithm.

In order to minimize the effect of light direction in the experiments readings were taken without any shadows within view. So light sources were never directly above the camera but at a slight angle so shadows fell outside the frame of view. This means that both the finger and background received the same amount of light.

All images were taken at a resolution of 320×240 and converted into a bitmap as input to the motion detection algorithm. All benchmarks were tested against a brief video (relative to each benchmark) from which 10 frames were extracted. Each frame was then run through the algorithm and the accuracy scores calculated for each. The score from each of the ten frames were averaged to obtain the mean accuracy score.

5.2.1. Light Level Study

A light level study was conducted with the aim of finding out what the average tungsten and fluorescent light level is in various rooms containing these two different types of lighting. Tungsten light is light emitted from an incandescent light bulb. Which is most commonly found in homes and areas which require less obtrusive bright lighting provided by fluorescent bulbs, hence the all of the light meter readings were taken in private homes and offices.

Fluorescent light tubes are found mostly in offices and public areas. They are becoming more popular in homes because of their efficient energy use. The majority of fluorescent readings were taken in public areas (eight of the ten). The two contrasting ways in which tungsten and fluorescent light is generated results in a different type and colour of light being emitted. Hence the reason for testing both separately. Ten different locations were chosen for each of the types of lighting.

CHAPTER 5 - ACCURACY, COLOUR AND LIGHTING

Fluorescent readings were taken in:

- Dark Environments
 - Parking garage
- Normal Environments
 - Three large retail stores
 - Two small retail stores
- Bright Environments
 - Three different corporate offices
 - Shopping mall

Tungsten readings were taken in:

- Dark Environments
 - Two rooms in three different homes (six rooms in total)
- Normal Environments
 - Three small offices
- Bright Environments
 - Winery tasting room

The light meter was situated directly under a light source and pointed upwards. The reading was then taken. This was repeated ten times in each location and the average mean calculated. The average values recorded for tungsten and fluorescent lit rooms are:

- **Average Tungsten Light Level:** 5 EV (± 0.1) ~ 80 lx
- **Average Fluorescent Light Level:** 8 EV(0.4) ~ 640 lx

Rooms lit by tungsten lighting tend to be much darker than those lit by fluorescent lights. This is in part due to the fact that multiple fluorescent tubes are used to light areas, while only single tungsten bulbs are used in homes and smaller venues. Fluorescent tubes also expose a much larger light emitting surface area. The light values of 5ev and 8ev were used for the tungsten and fluorescent respectively in the main accuracy study.

Table 5.2 shows the average readings for both sunlight at three times of day, fluorescent and tungsten.

Table 5.2: Average values recorded in the initial lighting study for the three different lighting conditions.

Dawn and Dusk	Mid Morning and Afternoon	Midday	Fluorescent	Tungsten
5EV	11EV	15EV	8EV	5EV

5.2.2. Backgrounds

The three common backgrounds chosen for the study are:

- Complex: A scene involving many different objects and colours. E.g. a scene involving many objects and edges (Figure 5.1 a)
- Texture: A scene involving a repeating object such as grass (Figure 5.1 b)
- Simple: A background which consists of predominantly one colour e.g. concrete (Figure 5.1 c)

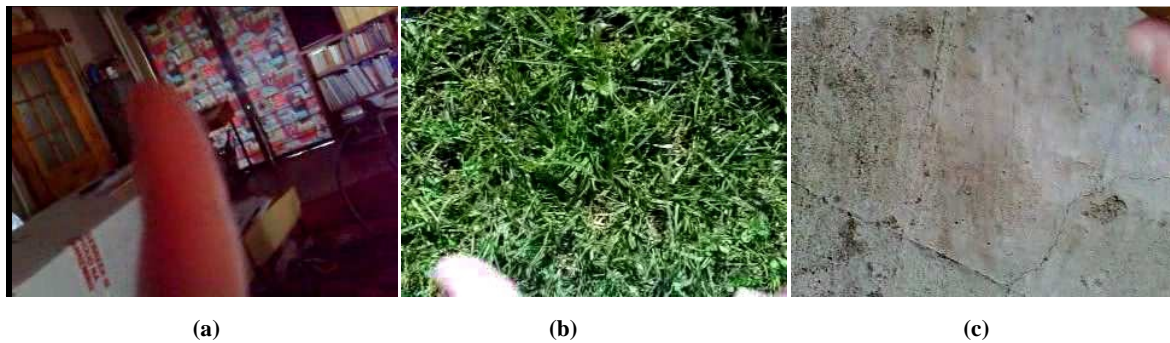


Figure 5.1: Examples of a complex scene (a), textured scene (b) and simple scene (c)

Mobile phones can be used in most environments, and hence testing the system cannot be constrained to a single lighting level and environment. Therefore the three different backgrounds chosen were chosen because they exhibit various general characteristics which are present in other environments. For instance the textured background (grass) contains many edges and layers which would be found in various natural environs such as layers of leaves, a field or a carpet. Another example is the simple background (concrete) which exemplifies many characteristics of uniform manmade and natural surfaces, such as dirt roads, asphalt or slate. The complex background represents a difficult background for the algorithm as it includes

CHAPTER 5 - ACCURACY, COLOUR AND LIGHTING

many different patches of colour and light levels. It is with these three backgrounds and five different light levels (dark sunlight, normal sunlight, bright sunlight, fluorescent and tungsten) that many of the problems likely to be experienced due to background colours and light levels can be identified. To further study different conditions some special cases were identified which were perceived to possibly cause accuracy problems for the algorithm (finger on similar coloured background, walking on textured background and moving shadows).

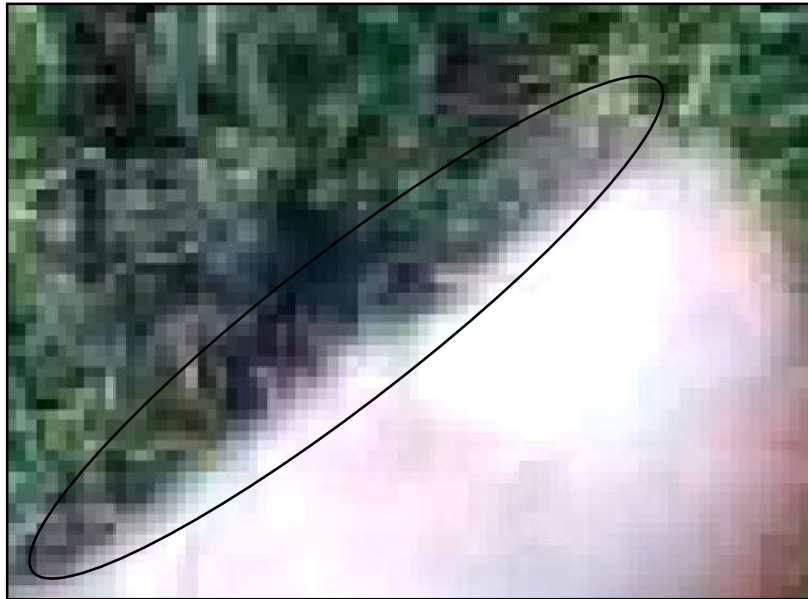


Fig 5.2 An anti-alias effect formed between the background and foreground objects

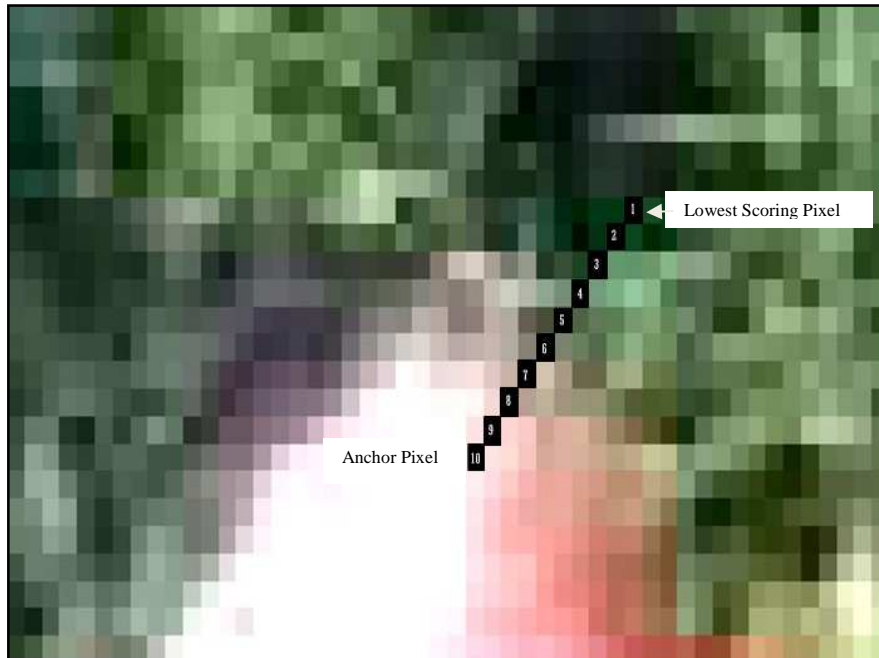
A side effect of the video feed which translates into the viewfinder feed and the frames received by the algorithm is an anti-aliasing effect which occurs around objects. This is demonstrated in figure 5.2 with colours from the finger washing into the colours of the background (encircled in the figure). If the viewfinder algorithm is not set up to strongly detect the finger colour then false positives may be generated from this effect (which would show up first because of the left to right scanning). To reduce any false positives a colour which contrasted highly with the background was chosen and calibrated to be tracked in the videos.

5.2.3. Accuracy Measurements

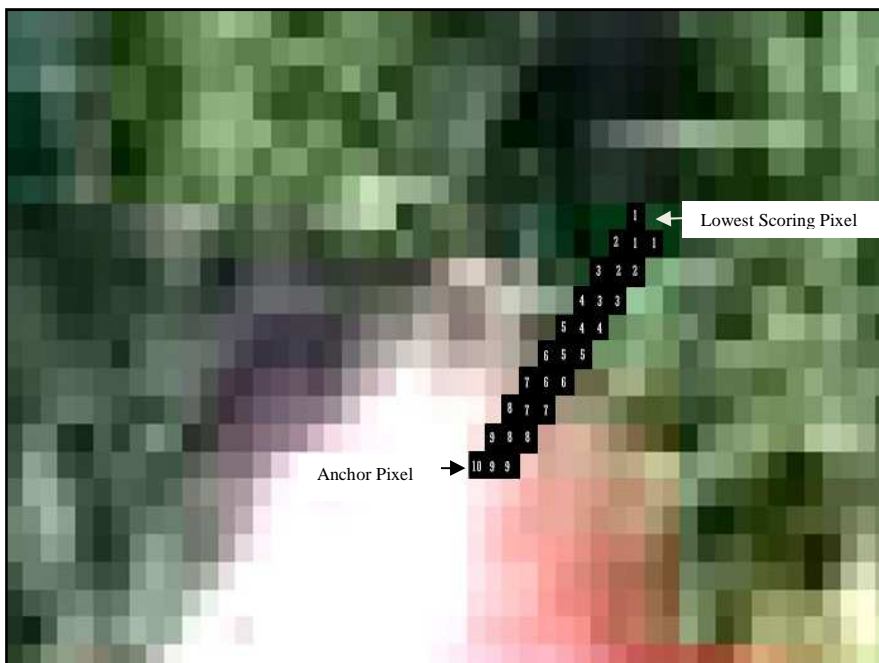
There were no examples in the literature of this type of accuracy experiment. Hence one had to be designed which could measure the accuracy of this system. This section explains the rating system that was used to calculate the accuracy score. Once the various frames were passed through the algorithm and the results analyzed for the position of the finger this rating system was applied. As shown in Section 3.3.1 the bitmap stream is obtained from the camera with each pixel's red, green and blue properties being examined for

CHAPTER 5 - ACCURACY, COLOUR AND LIGHTING

similarity to the reference colour. A point on the tip of the finger delegated to be the first section of continuous calibrated finger colour was used as the anchor pixel which should be detected. Pixels further away from this anchor pixel were assigned lower values (Figure 5.3 a).



(a)



(b)

Figure 5.3 Ranking system or accuracy values (10 is highest rank and 0 lowest)

CHAPTER 5 - ACCURACY, COLOUR AND LIGHTING

Figure 5.4 shows two pixels. The blue pixel is the very tip of the finger and can be seen as the goal or anchor pixel for the algorithm. This is the pixel which was used to calibrate the algorithm hence this pixel's colour is exactly the same as the reference colour. If the algorithm detects this then 100% accuracy has been attained. However if the algorithm detects the tip of the finger as pixel B (yellow) then a lower accuracy has been achieved based on the distance of this pixel from Pixel C. Pixel A (red) has been detected on the finger but is still 4 pixels away from C. Pixel A will receive a ranking of 10 (the maximum possible) – 4 (distance to the tip of finger) = 6 (final accuracy score). The pixel ratings can be seen in Figure 5.3 a and b.

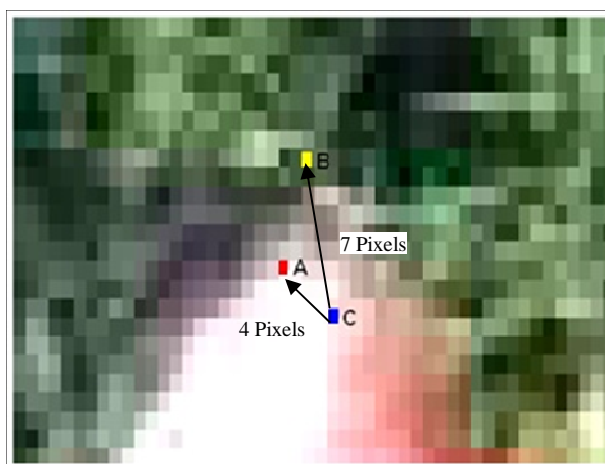


Figure 5.4: Pixel C represents the tip of the finger. Pixel B is a bad detection and results in an accuracy score of 3. Pixel A is on the finger but 4 pixels away from C and hence results in a score of 6.

5.2.4. Special Cases

Three special case scenarios were also recorded using backgrounds which were perceived to possibly cause a drop in accuracy. The first was of a very uniform colour similar to that of the finger, which produces a low score. The second was of a moving textured background generated by walking with the mobile phone while filming the tracking sequence. The third was filmed on a background which was perceived to cause problems because of the high contrast between sections of the background which were in shadow and sections which were in direct sunlight. The similar background special cases was filmed at the fluorescent light level measured in the light level study of 8 EV. The two other special cases were filmed at the Normal sunlight light level of 11EV.

5.3. Results

This section presents the results obtained from the various recorded scenes. The results are presented in Table 5.3 followed by a discussion on their significance.

Table 5.3: Accuracy ratings (mean \pm SE) based on distance from top most tip of finger.

	Dark(5EV)	Normal(11EV)	Bright(15EV)	Fluorescent(8EV)	Tungsten(5EV)
Complex	5.4(\pm 0.3)	5.9(\pm 0.3)	6.4(\pm 0.3)	6.6(\pm 0.4)	5.2(\pm 0.2)
Texture	6.6(\pm 0.3)	7.1(\pm 0.4)	7.6(\pm 0.4)	7.6(\pm 0.4)	6.2(\pm 0.3)
Simple	7.2(\pm 0.4)	7.6(\pm 0.4)	8(\pm 0.4)	8.4(\pm 0.4)	7.3(\pm 0.3)

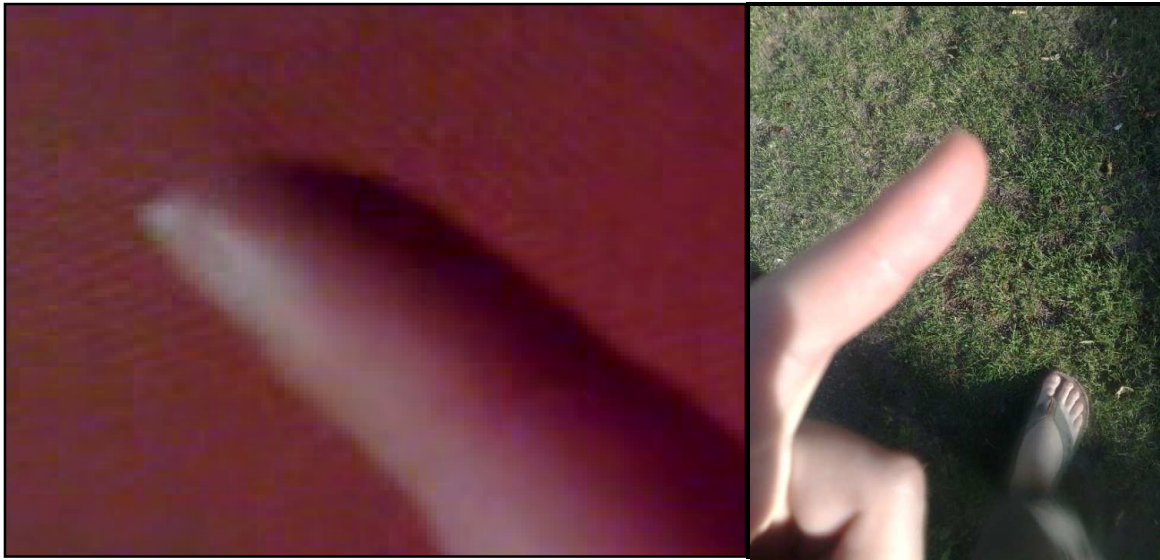
A one-way ANOVA with a Tukey post-hoc test was run on each of the data obtained for each of the backgrounds. On a complex background there was a significant difference in accuracy scores when moving from a dark (MS = 0.48222, $p = 0.019288$) or tungsten (MS = 0.48222, $p = 0.003254$) lighting condition to a bright or fluorescent lighting condition. No other significant differences within the complex background data were found. The textured background data showed the same pattern as above with a significant difference being detected between dark (MS = 0.61556, $p = 0.049078$) and tungsten lighting (MS = 0.61556, $p = 0.002248$) and bright and fluorescent lighting conditions. No other significant differences within the textured background data were found. The ANOVA test for the simple background showed that there was a significant difference between dark (MS = 0.36667, $p = 0.038049$) lighting and bright and fluorescent (MS = 0.36667, $p = 0.000665$) lighting conditions. Tungsten (MS = 0.36667, $p = 0.001833$) only showed a significant difference with fluorescent lighting. These results show that significantly darker lighting lead to a significant decrease in accuracy. Similarly significantly brighter levels of lighting lead to more accurate readings. Changing background composition has a similar effect with accuracy decreasing the more complex and colourful the background becomes. Combining dark light levels and complex backgrounds leads to very low accuracy scores where the interaction technique becomes unusable.

It is also prudent to note that there are some special cases where the lighting does not always increase the accuracy, or has little effect. These cases are: a finger on a similar colour background, Finger in moving shade on textured background and moving while using the algorithm on a textured background.

5.3.1. Special Cases

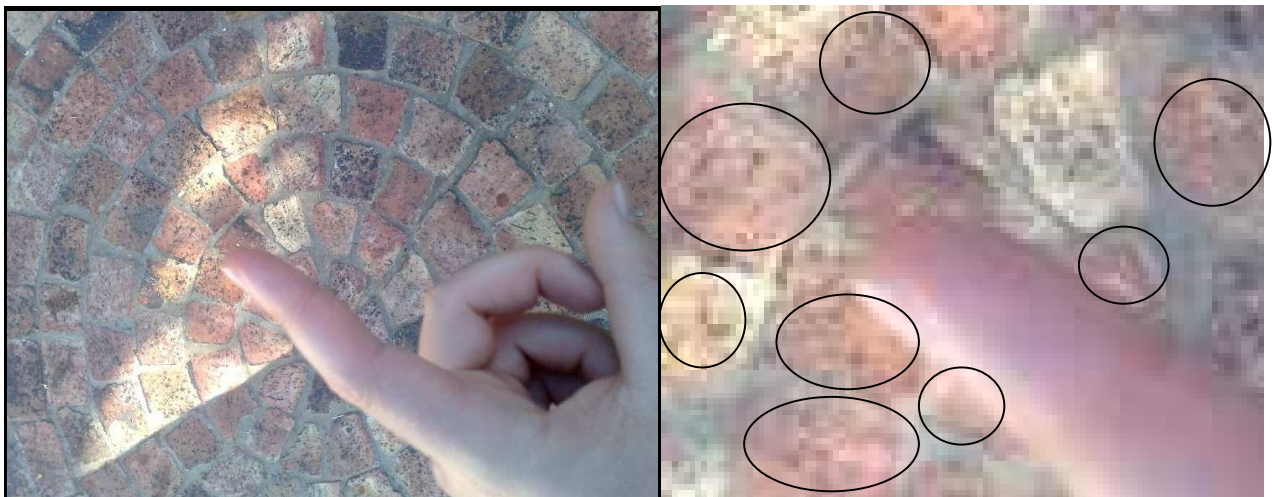
The special case results are:

- Simple on similar background: 2.3
- Walking on textured background: 5.4
- Moving Shadow: 4.5



(a)

(b)



(c)

(d)

Figure 5.5: (a) shows a frame with the finger being tracked on a similar colour background. (b) Shows a frame taken while walking on the textured background (grass). (c) shows a frame taken while walking on bricks to test a background with both a similar colour and moving shadows. (d) shows a close up of the tip of the finger problem areas (similar coloured areas) highlighted

CHAPTER 5 - ACCURACY, COLOUR AND LIGHTING

The special cases represent environments which pose difficult situations for the algorithm. It can be seen in Figure 5.5 (a) and (d) that subjecting the algorithm to a scene where the background is of a very similar colour to that of the finger proves to be a problem and results in a low accuracy score. The finger quickly gets lost in the background as consecutive sections of the background are incorrectly identified as the finger. The pointer is placed in the top left where the algorithm begins scanning and stays there as the colour does not change. Walking on a textured background gives better results as long as not too many colours of similar values to the calibrated colour move into the scene. Figure 5.5 (b) shows a frame taken whilst walking on grass. The shifting background does not pose too much of a problem for the algorithm. The foot in view in the lower left is of the same colour as the finger. However it is below the finger and therefore will not be detected as the finger is detected first, secondly the foot is not of consistent colour as that of the finger and hence will not be detected.

5.3.2. Analysis

This section presents an analysis of the results, focusing on behavior of the algorithm when presented with the various colours generated by the different conditions.

5.4.1.1. Bright

A finger in direct bright sunlight reflects a large portion of the light away from itself, creating a large portion of very light colour (Figure 5.6 shows the finger as mostly white in colour). The finger gets lost quickly if the lighter colour is tracked. Taking advantage of the darker part of the finger produces a 100% accuracy on this plainly textured background.

Tungsten: General tungsten lighting tends to be quite dark producing a lower ambient light. Because of the general low light level of tungsten there is also a lower accuracy under tungsten lights than there is under brighter fluorescent lighting and sunlight.

Direction of light plays a large role in accuracy. A finger which highlighted by the light source with the background in shadow makes finger easier to track. However if the background is lit and the finger is in shadow the finger is lost in the vivid bright background and is harder to track.

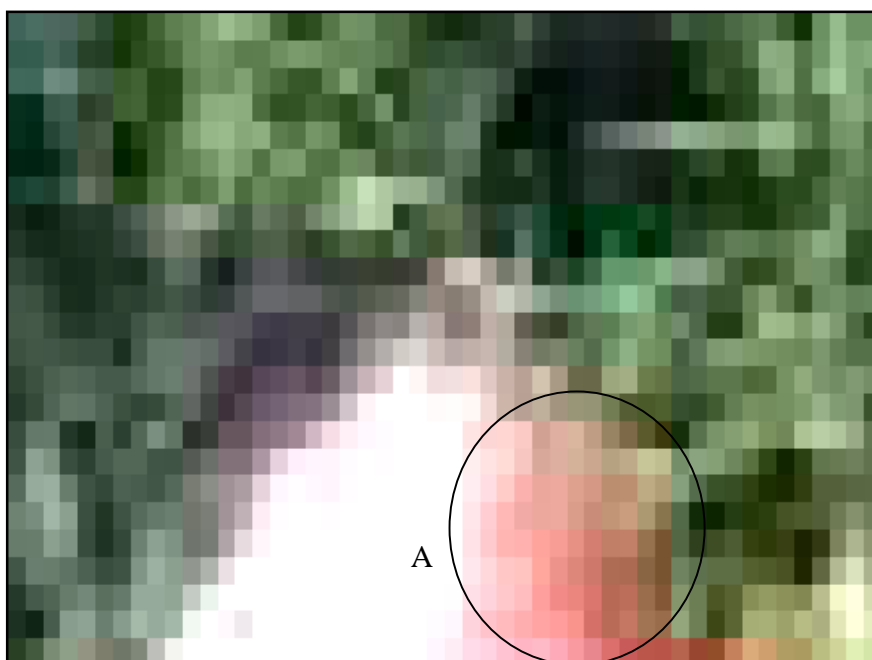


Fig 5.6: Close up view of a frame with the grass and finger at bright light levels. Note the majority of the finger turns completely white due to the bright overhead sun. The red part of the finger is highlighted in A

As can be seen in Figure 5.6 there are a few patches of white which may show up as false positives if the white part of the finger is used as the reference colour. However if the more red part of the finger is used (bottom right part of finger highlighted in A) then a much higher contrast between the light pink and predominantly green background can be used for easier detection.

However as can be seen in Figure 5.6 almost all of the finger is washed white with a small bit of red occurring further down. If only the red part of the finger was being tracked then the algorithm would not detect the tip of the finger Here it can be seen that an average of both the light red and the white will account for more instance than simply one or the other.



Figure 5.7 Finger shown on similar colour background

CHAPTER 5 - ACCURACY, COLOUR AND LIGHTING

In this case the finger was placed in front of a surface which was of a similar colour note that the more red parts of the finger are almost identical to the overall colour of the background. The only section which consistently stands out against the background is the white sheen on the tip of the finger caused by the overhead lights, shown in greater detail in Figure 5.7 (b) and highlighted in (a). It would be much more sensible to use this off white colour as the reference colour than the redder part of the finger for ease of tracking.

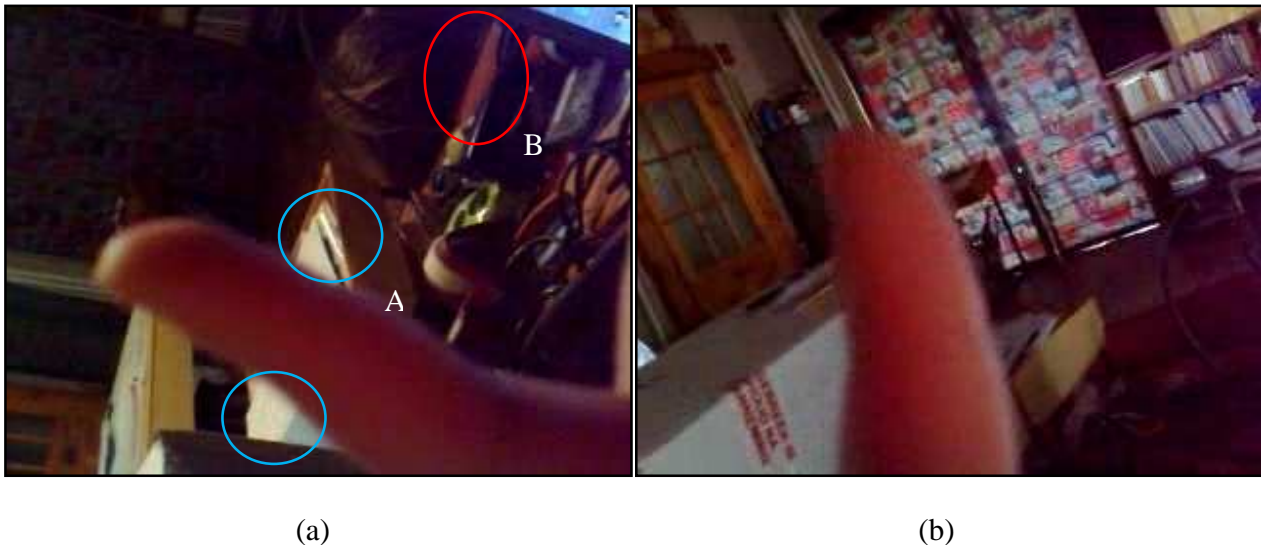


Figure 5.8: (a) shows a complex scene under low Tungsten light. Problem Areas are highlighted (White areas in light blue, light red areas in red). (b) shows the same complex scene but with ambient sunlight illumination

Figures 5.8 demonstrates two of the most prominent problem areas with a complex scene. Portion A shows problems for the white sheen section of the finger whereas section B shows problems for the red area of the finger. Figure 5.8 (b) shows the difference in colour when a different lighting condition is used. In this case sunlight.

5.4. Summary

This chapter presented various lighting and environmental factors under which the finger tracking system was put to the test. All scenes were prerecorded to minimize the variance between scenes and provide a benchmarking tool for future developers of visual interactions on mobile phones. All scenes were run through the thresholding algorithm and their results based on a ranking system were recorded. Some interesting case studies were also presented and explored, exhibiting key situations in which the performance of the visual interaction system would be less than satisfactory. It was found that fairly high levels of lighting are required for the interaction technique to be of any use. A direct relation between

CHAPTER 5 - ACCURACY, COLOUR AND LIGHTING

background complexity and accuracy was also found with complex busy backgrounds yielding the lowest accuracy results and simple backgrounds yielding the highest. These results have quantified the way in which the interaction system is affected by the environment in which it is used. This is important when comparing it to the standard keypad layout which is not as adversely affected by dark light levels (using backlights) and not affected at all by the environment in which it is used. The actual relationship between these two different types of interaction techniques is explored through a user study in the next chapter.

Chapter 6 - User Study

A user study was conducted in order to ascertain the usability of the finger interaction technique as well as the user's satisfaction by obtaining qualitative feedback. First an introduction to mobile usability is presented followed by an explanation of the performance metrics and user tasks. Three applications using both the finger interaction technique and keypad were developed. These were created to assess mouse based interactions, steering based tasks and gesture interactions. These will be introduced and discussed in section 6.3, followed by a methodology of the user study. Finally a discussion of the results is presented.

6.1. Introduction

Mobile applications must be usable and useful, they must also provide a consistent and comprehensive user experience (Jones & Marsden, 2006). To be useful and usable an application must allow users to easily work out how to complete their task. For instance a user who wishes to browse a webpage should not be hindered by the interface presented to them by the browser. To be consistent and comprehensive an application should use the same method of interaction throughout. An application should not switch between directional keys and number keys for reasons unknown to the user. This would create confusion in its lack of consistency.

Mobile phone applications are designed to take advantage of the current keypad layout (Wisniewski, et al., 2005). However the current keypad is designed with menu navigation and number/text entry in mind. This does not provide adequate support for steering tasks, gestures and mouse interactions. The underlying hardware of mobile phones is increasing in power and hence so must the interfaces that govern them (Kerr, et al., 2008). The finger interaction technique hopes to address the problem of only keypad support for the majority of mobile phones. To ascertain the affect this interaction technique has on steering, mouse and gesture based tasks three applications were created which assessed the usability and user satisfaction of the finger interaction technique.

It is important to test the user experience on mobile phones as this is what governs the way in which users interact with them (Jones & Marsden, 2006). Norman (1988) lists four principles which make a system easy to use, these are: visibility, feedback, a good conceptual model and good mappings. A high degree of visibility allows the user to conceptualize the current state of the application and the range of actions which are possible. Constant feedback must be provided to the user so they know what the results of their actions are. A good conceptual model allows the user to understand the relationships between all the different parts

CHAPTER 6 – USER STUDY

of the system. Good mappings create rational relationships between the actions the user performs and the results they receive. The user study applications were designed to assess the usability of the finger interaction technique. One could relate the four principles to the finger interaction technique thus:

- **Visibility:** The system has three noticeable modes: pre calibrated, calibrating and calibrated. This allows the user to know what mode the system is in and what input it is expecting.
- **Feedback:** The user is presented with the position of their finger which updates with each frame.
- **Conceptual model:** it is clearly shown to the user when their finger is being tracked and when objects on the screen have been interacted with. This allows them to understand the states the system is in.
- **Good mappings:** Mapping the output to the finger allows for a direct relationship between user movement and the positional information on the system.

All these principles aim to improve the usability of the system. But this must be evaluated in a user study which compares the usability of the finger interaction technique to the keypad interaction technique. As such a series of performance metrics were designed to test the tasks.

6.2. Performance Metrics

Love (2005) suggests three performance metrics when evaluating mobile systems, these are: task success, task completion time (time on task) and errors. Hence these three performance metrics were used in the study. These are: Task Success, Time-on-Task and Errors. Each of these metrics will be discussed in the following sections.

6.2.1. Task Success

Task success is one of the most widely used performance metrics (Love, 2005) and measures the ability and effectiveness of the user to complete the task given to them. Two types of task success exist, namely *binary task success* which is a simple affirmative or negative score awarded when the user completes the task, or *levels of success* which adds grey areas which define how well the user managed to complete the task. This study identifies different levels of success to take into account various situations in which a user may have partially completed a task This study used overall binary task success in that users either succeeded or failed the task.

6.2.2. Time-on-Task

Time-on-task measures the length of time the user takes to complete a set task. The faster a user completes a task the better the experience for them. However a number of exceptions do exist, such as some games would not be enjoyed to their fullest if the user simply rushes through them (Wisniewski, et al., 2005). However in the case of the steering game a faster time is in fact better for acquiring a high score as it is a reflex game relying on the user's quick finger movements.

Time-on-task is important for this study as it measures whether or not the visual finger interaction technique is faster than normal keypad input. All applications were timed starting from after the user had pressed the OK key and the system had calibrated to track their finger colour (the pointer is drawn at this point and is the signal for the user to begin and the timer to start), to when they attained the score in the steering game, acquired all the targets in the target acquisition application or completed the series of gestures in the directional gestures application.

6.2.3. Errors

Errors reflect mistakes which are made during the tasks. They are useful in highlighting problem areas of an interface and were used to great effect in this user study. In this study, errors were broken into two groups: system errors and user errors. System errors are errors caused by the system losing the finger in the background and hence obscuring an accurate motion. User errors are errors caused by the user making a mistake. System errors are the same across all the applications but user errors differ between the applications. Two types of user errors are common to all of the applications, these are: calibration errors and out of view errors

Calibration errors were recorded if the user held their finger in the incorrect place and initiated a calibration request, hence inputting a part of the background or incorrect part of the finger. Out of view errors (OVE) were recorded if the user moved their finger out of the view of the camera hence causing it to lose the track of the finger and giving a false positive result or no result at all to the application. It should be noted that the users were told not to move their fingers out of view and an OVE was recorded by the system on the phone.

6.3. Tasks

Three applications were created whose subtasks were based on Foley *et al's* taxonomy of subtasks (as discussed in Section 2.3.4). These applications were created to test the usability of each type of interaction in each task and acquire feedback on the new finger interaction technique. All applications were equipped

with the ability to accept both key presses from the keypad and coordinate updates from the camera in order to compare the standard keypad interaction technique with the finger based interaction technique. The three different applications are discussed in the following sections.

6.3.1. Steering game

A steering based game was developed to test the system in relation to quick reaction based movements which are seen in many mobile phone arcade style games such as Snake on Nokia phones (Nokia, 2003). This task falls within the position subtask of Foley's taxonomy in that the user must move the pointer to catch the falling blocks. This requires the user to position the pointer in one dimension to catch the blocks. The game involves the user moving their finger to catch objects falling from the top of the screen. When an object is caught it changes colour, is reset and the score increased. Figure 6.1 illustrates the game being played using the finger interaction technique. The user has caught three of the targets which have changed their colour to white and are no longer worth points. The user must now try and catch the left most target to reset all the targets. The right most target has reached the bottom of the screen and is in the process of being reset as shown by the faded yellow block at the top right of the screen.

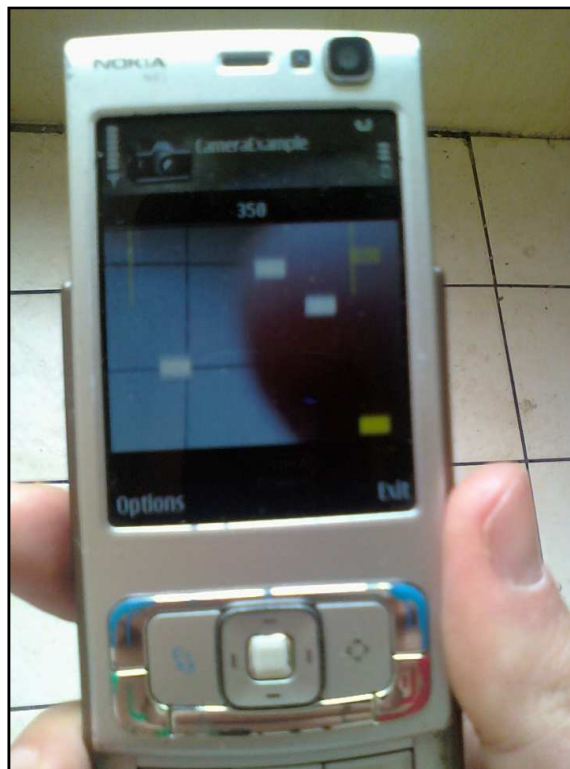


Figure 6.1: User playing the steering game. The application is tracking the darker part of the finger and is hence tracking the lower part of the finger on the screen.

CHAPTER 6 – USER STUDY

To succeed in the steering game the user must complete the game. In order to carry this out the user must catch 15 blocks. Once they reach the designated score the task is ended and a success (1) is recorded. If the user gives up, a failure (0) for the task is recorded.

The internal timer started when the pointer was drawn on the screen, after the calibration process. Once the required number of blocks were caught (or score was achieved) as specified in the task brief (see Appendix A) the timer was stopped and the log of the time and error values written to a file.

User errors in the steering game were limited to only calibration errors and out of view errors as a user cannot be penalized for slow reflexes or loss of concentration. However the number of blocks missed by the users were counted for both the keypad and finger interaction technique runs on the task. This provided a comparable way of recording how many errors were performed for each interaction technique.

6.3.2. Target Acquisition

This application was designed to test the navigational ability of the finger interaction technique. This task falls within path and select sub tasks. The user must move (create a path) the pointer in two dimensions (an important difference from the steering game and directional gestures task) from one block to the other in a given order. The user must also acquire each block to move on. By moving and acquiring blocks the user is emulating acquisition, whether it be in menus, internet browsers or applications. Four Targets are placed on the screen at equal distance from each other. The user must then move their finger to acquire each target on the screen by hitting the target with the pointer in a clockwise order. Figure 6.2 shows the target application in progress.

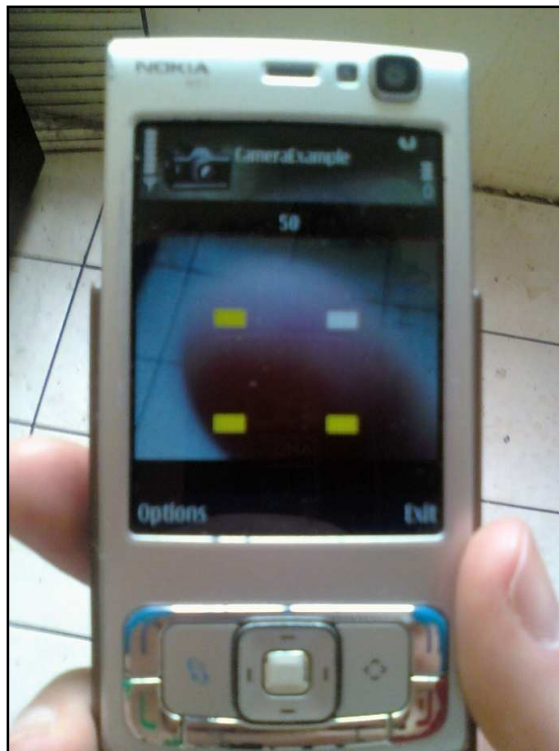


Figure 6.2: User using the targeting application. The user is moving their finger away from the block they acquired in the top right shown in white. The three yellow blocks have not been acquired by the user.

To record a success users must use their finger or navigate with the keypad to acquire four equally spaced targets on the screen. Once all targets have been acquired a success is recorded. Figure 6.3 shows a users path through the target acquisition application. The user starts by selecting the top right block. They then move in a clockwise order to select the bottom right, bottom left and finally top right. The user does not have to carry out all these selections in one clockwise movement. They can move to and from the centre if they wish, as shown by the red path.

To succeed in the target acquisition task the user must complete the required amount of selections in the correct order. Once they reach the designated number of selections the task is ended and a success (1) is recorded. If the user gives up a failure (0) for the task is recorded.

The internal timer started after the user pressed the OK button to calibrate the colour of their finger. Once the required number of blocks specified in the task brief (see Appendix A) were selected the timer was stopped and the values written to a file.

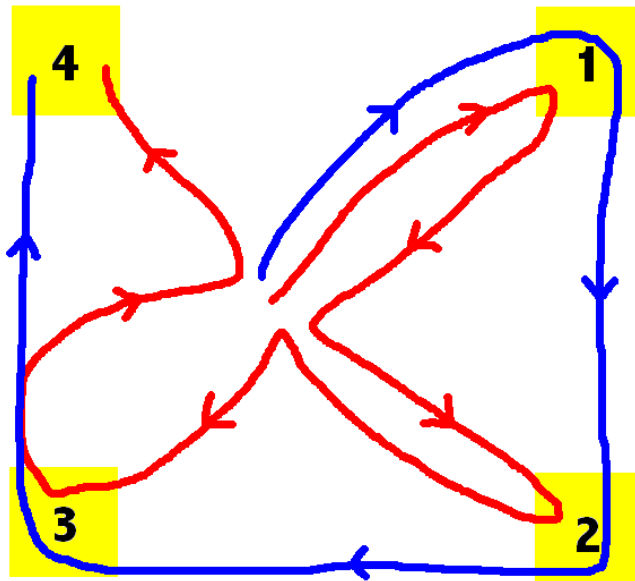


Figure 6.3: A sample user's finger path using the target acquisition application. Starting with the top right target the user selects all target in a clockwise order. Both the red path and blue path are considered successes

Additional errors were recorded if the user acquired the targets in an order different to the one stated in the task brief. These are classified as user errors and as such were not errors caused by the system but errors caused by the user's lack of concentration in understanding or completing the task. The main errors related to the interaction technique are the amount of reattempts the user needs before the block is registered as selected (see figure 6.4). If the user misses a block and needs to reverse direction to acquire it then this is recorded as a reattempt. However if the user pauses or moves their finger back upon itself when not close to a block and hence is not trying to select a block then this is not counted as a reattempt.

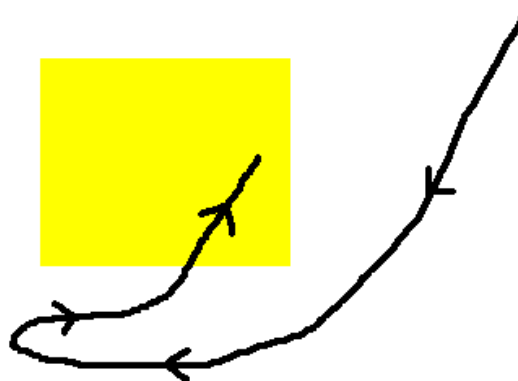


Figure 6.4: An example of a reattempt at acquiring a block. The user has overshoot the block and had to backtrack

6.3.3. Directional Gestures

An application was created to test simple gesture based interactions. Directional gestures are gestures used on touch screen devices (Lin, et al., 2007). These gestures also emulate the way in which the user would use current keypad based interaction techniques on the mobile phone, such as scrolling through a set of pictures, or scrolling down a menu. This task falls within the pathing, select and gesture (Reimann & Paelke, 2006) subtasks of Foley *et al's* (1990) taxonomy because the user must utilize finger gestures either from left to right or right to left in order to move the blocks. Smaller gestures to the middle of the camera's field of view select the blocks. The user is presented with a screen containing a series of coloured blocks. In order to move to the next block the user must move their finger from the left of the screen to the right to move to the right hand block, or from right to left to move to the left hand block. Figure 6.5 shows the user with their finger to the right of the screen and the blocks scrolling in that direction. Similar to the steering game the user is limited to one dimension. Unlike the steering game the user must also take into account selection and pathing of the gestures. The internal timer started when the user pressed the OK key to calibrate the colour of their finger. Once the sequence of blocks specified in the task brief (see Appendix A) had been completed the timer was stopped and the time and error values were written to a file. Reattempt errors were monitored by an observer, all other errors were recorded internally.

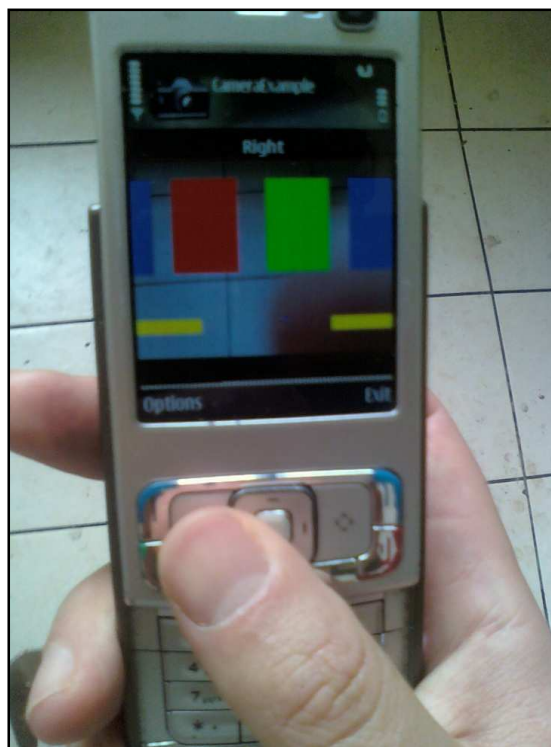


Figure 6.5: User using the directional gestures application. The user is moving the blocks to the right

CHAPTER 6 – USER STUDY

Task success (1) was recorded if the user completes the designated number of block selections. Failure (0) was recorded if the user gives up.

Additional errors in the directional gestures application were recorded if the user motioned a gesture at the wrong angle. For instance the task was to use the finger to move a set of coloured blocks either left or right. If the user moved their finger in a vertical motion instead of horizontal then this was considered as a user error. If the user overshoot a box they were trying to select then an overshoot error was recorded. Figure 6.6 shows the user having to backtrack to select a square.

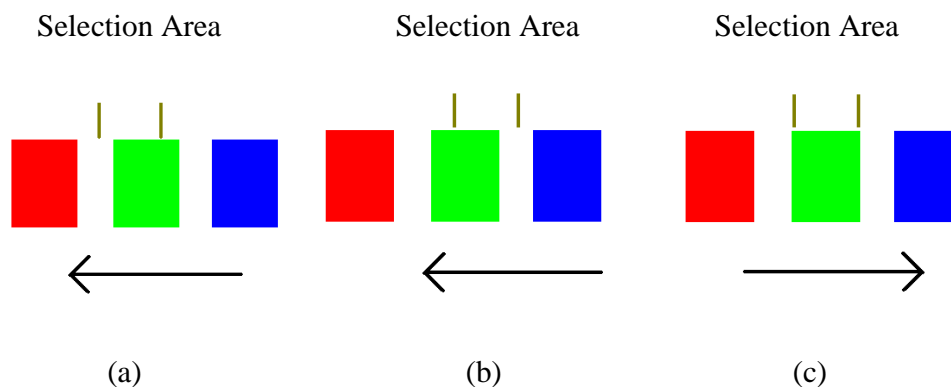


Figure 6.6: In (a) the user is moving the blocks left with the intension of selecting the green square by making it come to rest between the two lines. (b) shows the user over shooting the lines and having to backtrack to the right to finally select the square in (c)

6.4. Methodology

This study used a repeated measures design (within-subject design) which means that all participants were tested on both interaction techniques (Love, 2005). In order to negate order effects arising from practice (i.e. user's becoming familiar with the system after their first use and hence biased results are achieved for the second interaction technique) the sample of users was divided into two groups using a Latin Squares Design (Cairns & Anna, 2008). Table 6.1 displays the way in which two users were divided into two groups and tasks assigned to them. One group was started on the keypad interaction technique and the other on the finger interaction technique.

CHAPTER 6 – USER STUDY

Table 6.1: A Latin Square with three different tasks. This is an example of two users, one in group i and one in group ii. A random selection of tasks is presented.

Group	First Task	Second Task	Third Task
i (Keypad First)	Steering Game(Keypad)	Target Acquisition (Keypad)	Directional Gestures (Keypad)
	Steering Game(Finger)	Target Acquisition (Finger)	Directional Gestures (Finger)
ii (Finger First)	Target Acquisition (Finger)	Directional Gestures (Finger)	Steering Game(Finger)
	Target Acquisition (Keypad)	Directional Gestures (Keypad)	Steering Game(Keypad)

The independent variable is the amount of errors performed by users. Outcomes caused by the independent variable, known as dependent variables are twofold. Firstly the user’s speed of performing the tasks is dependent upon their comfort and ability with the finger interaction technique. This is the first quantitative data that is recorded. Secondly the user’s attitude and overall satisfaction with the finger tracking interaction technique is in question and their responses are recorded to ascertain the overall acceptance of the new interaction technique. It is important to note that this second dependent variable should not be underestimated in its importance. While user’s may perform some tasks much faster using the finger interaction technique, they may not be comfortable using this technique and hence would not want to use it. This is especially true of new interaction techniques on mobile phones as the user will often be using the device in a public area, sometimes on the move and may feel that the workload is too high.

6.4.1. Apparatus

This study required:

- The development of three applications on the mobile phones using the two different interaction techniques
- Creation and setup of the speed timings for the interaction techniques. These were recorded with internal timers.

CHAPTER 6 – USER STUDY

- Development of the questionnaires which would gather the information after the tasks were completed (see Appendix A). These give the user's experience and how they felt when using the system.

Two questionnaires were developed. The first was based on the NASA – Task Load Index (TLX) (Hart & Staveland, 1988) which is designed to gather information after each task was completed. This questionnaire gathers information based on six categories of task load, these are: mental demand, physical demand, speed, success, work load and difficulty. It consisted of six questions, each addressing one of the six task load categories. Below each questionnaire was a selection for the user to specify which technique they preferred (Finger or Keypad) and a space for any further comments.

The second questionnaire was based on the Usefulness, Satisfaction and Ease of Use (USE) questionnaire (Lund, 1998). This questionnaire was chosen because it analyzes usability metrics and not performance metrics like the TLX questionnaire. Three System Usability Scale (SUS) questions were also migrated into the USE questionnaire into the relevant sections, based on the results shown in Tullis and Stretson (2004). The USE questionnaire contains four distinct categories, these are: usefulness, ease of use, ease of learning and satisfaction. Usefulness included questions on the desired efficacy of the finger interaction technique and whether the user thought the technique would save them time. Ease of use included questions on the user friendliness and whether the user could use it without instructions. Ease of learning included questions on how quickly the user learned to use it and whether they thought other people would be able to learn it easily. The satisfaction category included questions focused on how fun and pleasant the interaction technique is. Questions were ranked using a 7 point Likert Scale. There were 7 questions in the usefulness category, 11 in the Ease of use category, 4 in the Ease of learning category and 6 in the Satisfaction category, with a maximum attainable score in each section of: 49, 77, 28 and 42 respectively (higher is better). Those questions which were not positive had their scores inverted when they were tallied. Each category section was tallied and converted into a percentage for comparison. This allowed the four categories to be compared. This questionnaire asked questions only about the finger interaction technique.

The study required a mobile phone capable of performing the finger based interaction tasks and the Nokia N95 was used for the prototype applications. The study also required certain environmental conditions. These were in a room lit by fluorescent lighting at 8 EV and on a simple grey background as this was shown to be the environment in which the system gave the most accurate results (see Section 5.2.1).

CHAPTER 6 – USER STUDY

A suitable location was chosen and the lighting tested with a light meter to ensure the system would not fail due to it being too dark or the background environment too complex. The following section details the participants who were chosen to take part in the user study.

6.4.2. Participants

Numerous researchers have said how many participants should be used in a user study. Faulkner (2003) suggests 10 to show 95% of user errors. Tullis and Stetson (2004) conducted a study which used 123 participants to evaluate a pair of websites using five different usability questionnaires (SUS, QUIS, CSUQ, Microsoft Product Reaction Cards and their own questionnaire). Once completed they took random samples between 6 and 14 to analyze the percent of correct conclusions based on the full sample size. They showed that at 12 participants some of the questionnaires had peaked (SUS, QUIS) with the others only increasing by two to five percent at a sample size of 14. Hence twelve participants were chosen who were familiar with using mobile phones. Users were also used to interacting with applications or games on their phones and had at least five years experience doing so. Participants were of varying ages between eighteen and thirty which is the age group with the highest penetration of camera based mobile phones (Wirefly, 2008). Six male and six female participants were selected to give a full representation of users. The colour of the user's skin is important in relation to the background being used. A user with darker skin may have problems when using the system in a dark environment. Similarly a user with lighter skin colour in an environment with many reds, pinks and whites will cause the system problems in distinguishing the finger from the background (see Section 5.3.2). Hence an equal distribution (6 dark, 6 light) of people with differing skin colour was attained in order to achieve a representation of user's finger colours who would use the system.

6.4.3. Procedure

Participants were evaluated separately using a random ordering of tasks to negate any possible familiarity or practice between interaction techniques. The participant was first given a session brief which contained an explanation of the system, the goals of the user study and how to work the finger interaction technique and how to switch the application to using the keypad (see Appendix A). They were then asked to sign a consent form in accordance with the Rhodes University ethics guidelines (see Appendix A). The participant was then given a brief demonstration of each of the applications and shown how to calibrate the system to recognize their finger. The participant was then given a chance to briefly experiment with a practice application in order to become comfortable with the way in which they held their fingers in front of the camera. The practice application tasked the participants with moving their finger between two obstacles to

CHAPTER 6 – USER STUDY

draw a figure of eight and use the keypad to achieve the same goal. Users took on average two minutes on this preliminary task. Once the participant was comfortable with the finger interaction technique and could easily trace the figure of eight the evaluation began. Those who were chosen to use the finger interaction style first were given the phone with a random application initiated. For example assuming the applications chosen were: the directional gesture application first, steering game second and target acquisition third, the session would have continued thus: The directional gesture application was started and handed to the user. They were then asked to select the sequence of blocks outlined in the task brief. Once this was completed the application was changed to use the keypad and the same sequence of directional gestures were performed. The final results were time on task, amount of overshoot errors and system errors. These were either written to file by the system or noted by the observer. Once the task was completed the user was given the TLX questionnaire, preferred system and general comments for the task(see Appendix A).

The second application was the steering based game. The user was asked to catch 15 blocks (acquire a score of 1000). Their run through the game was timed and the number of blocks the users missed and caught were also recorded. Once the user had completed their first run in the game it was reset and the user was then asked to use the other interaction technique to again get a score of 1000. The final results of this test were the amount of catchable blocks missed, OVE and time on task. Once the task was completed the user was given the TLX questionnaire, preferred system and general comments for the task (see Appendix A).

The third and final application presented to the user was the target acquisition application. Users were asked to select the objects in a clockwise order starting from the top right. As the user moved their finger to each of the blocks it would change colour letting them know that it had been selected. Once the top left block was selected (the last block in the clockwise sequence) all blocks would reset to their default yellow and the user could continue. The user would continue with this process until they had completed it five times. Once this was completed the application was reset and switched to accept input from the keypad and the same sequence of five clockwise block acquisitions was performed. The final results captured by the observer was time on task, OVE and amount of reattempts at selecting a block. Once the task was completed the user was given the TLX questionnaire, preferred system and general comments for the task (see Appendix A).

Once all tasks were completed the user was given the USE/SUS questionnaire. Once this had been completed the user was asked if they had any other comments on the finger interaction technique, if so they were asked to write it down on the paper provided.

6.5. Results and Discussion

Each of the three tasks generated four sets of results. These are:

- The time taken on each task,
- The amount of misses and errors,
- TLX score based and
- which of the interaction techniques the participant preferred.

Task success is not presented as all users were successful in completing all of the tasks they were given. No difference was found between users with different skin colours and so will not be discussed in relation to the results..

This section will first look at the time it took users to complete each task, followed by the amount of errors generated by the user when using the keypad and their fingers. This is followed by the task questionnaire TLX results and the sub categories for each of the tasks. Finally the USE/SUS questionnaire results will be presented and discussed.

6.5.1. Time on Task

The Time on task data is presented in Figure 6.7. A paired t-test was performed to determine if there were any significant differences between the keypad and finger interaction data. No significant difference ($t = 2.02$, $p = 0.039$, $df = 11$) was found between the keypad and finger for the target acquisition task. No significant difference ($t = 1.77$, $p = 0.06$, $df = 11$) was found between the keypad and finger for the steering game. A significant difference ($t = 2.65$, $p = 0.02$, $df = 11$) was found between the keypad (27.8) and finger (46.02) interaction times for the directional gestures task.

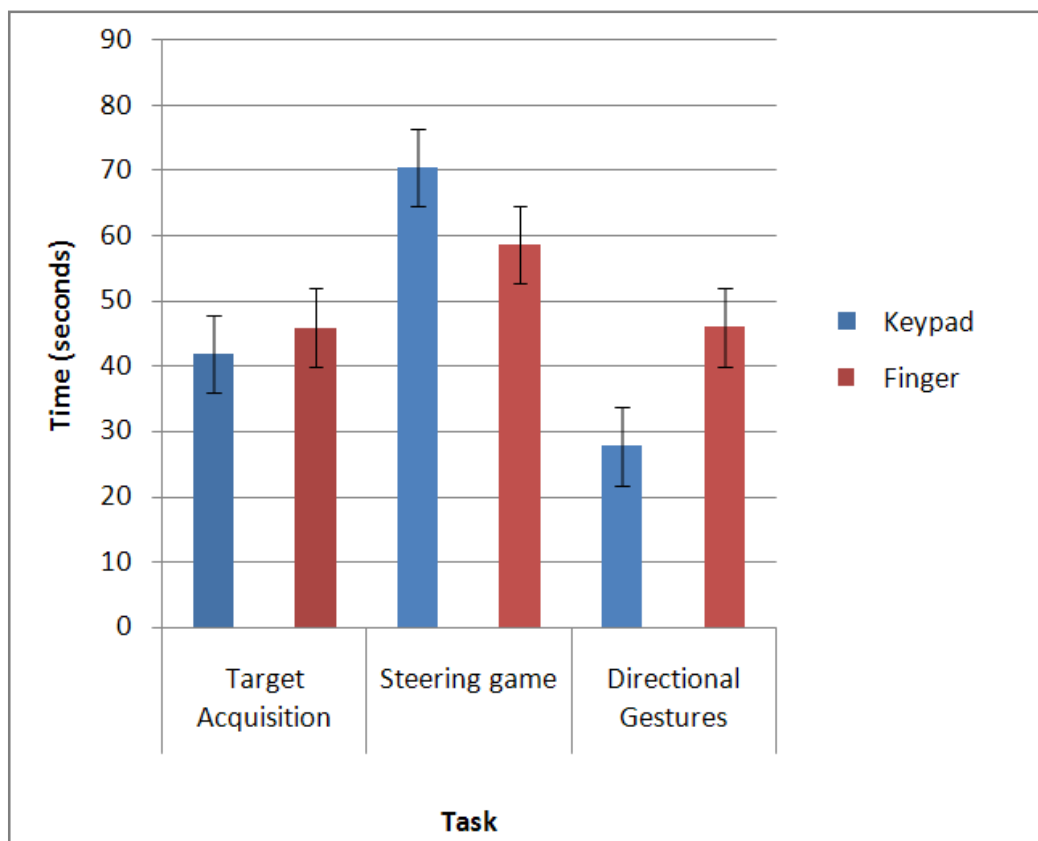


Figure 6.7: Time taken to complete each of tasks

6.5.2. Overall Errors

The overall errors are presented in Figure 6.8. User errors were not included. A paired t-test was performed to determine if there were any significant differences between the keypad and finger interaction data. A significant difference ($t = 5.61, p < 0.001, df = 11$) was found between the keypad (0.42) and finger (4.25) in the target acquisition task. No significant difference ($t = 2.004, p = 0.03, df = 11$) was found for the steering game. No significant difference ($t = 1.8, p = 0.05, df = 11$) was found for directional gestures.

As can be seen in Figure 6.8, in the target acquisition task users performed on average 0.4 errors when using the keypad. The finger interaction technique scored on average 4.25 errors per user. The amount of blocks missed in the steering game were counted and on average a user using the keypad missed 17 blocks whereas when using the finger interaction technique 14.25 blocks were missed on average. Errors recorded in the directional gestures tasks occurred 0.33 times per user when using the keypad and 2.92 times per user when using the finger interaction technique.

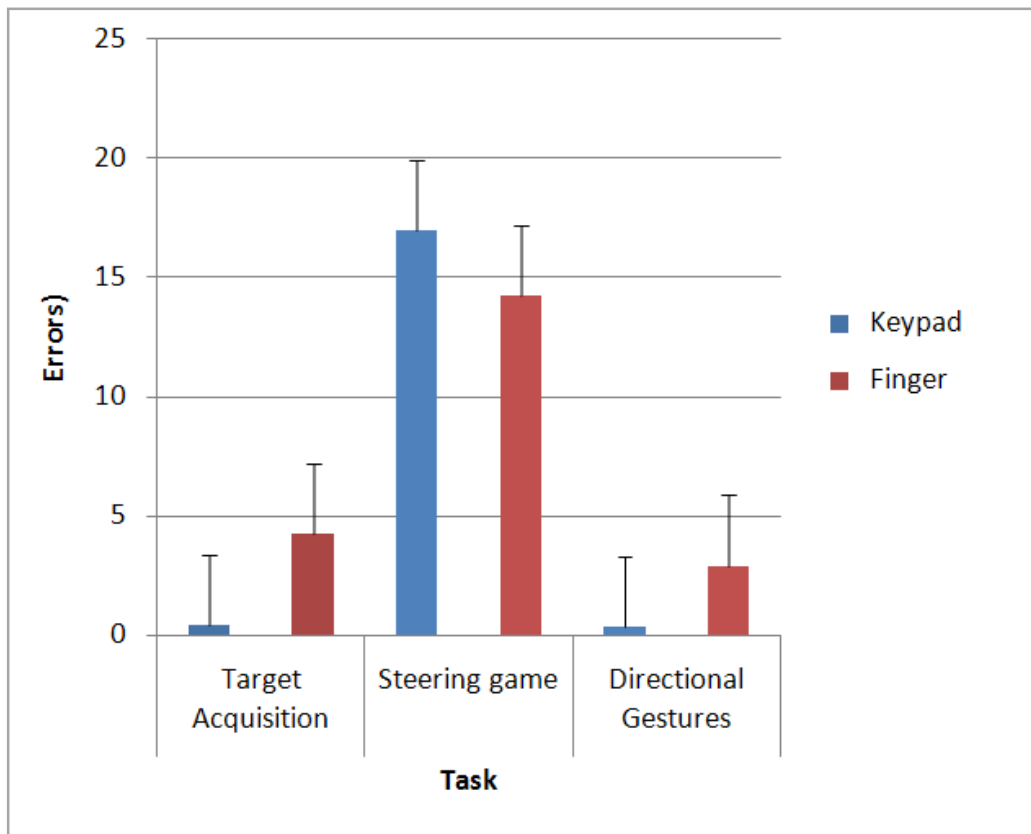


Figure 6.8: Number of overall errors within each of the tasks

6.5.3. TLX Score

The TLX score is presented in Figure 6.9. There were six questions, each out of ten with the maximum possible score being 60 (higher is worse). A paired t-test was performed to determine if there were any significant differences between the keypad and finger interaction data. A significant difference ($t = 2.63$, $p = 0.02$, $df = 11$) was found between the keypad (17.58) and finger (27.1) for the target acquisition task. No significant difference ($t = 0.7$, $p = 1$, $df = 11$) was found between the keypad and finger for the steering game. A significant difference ($t = 2.8$, $p = 0.008$, $df = 11$) was found between the keypad (15.5) and finger (27.1) interaction times for the directional gestures task.

Summing up the questionnaires gave scores for both the keypad and finger interaction techniques within each of the tasks. All questions were positive questions except for number four (the question score was inverted when tallied), hence a higher score is worse than a lower score. Figure 6.9 shows that users on the target acquisition task gave the keypad a better score than the finger interaction technique. The finger interaction technique scored similarly in the directional gesture task. However the finger interaction technique did achieve the same score in the steering game as the keypad, which is an improvement when compared to the other tasks.

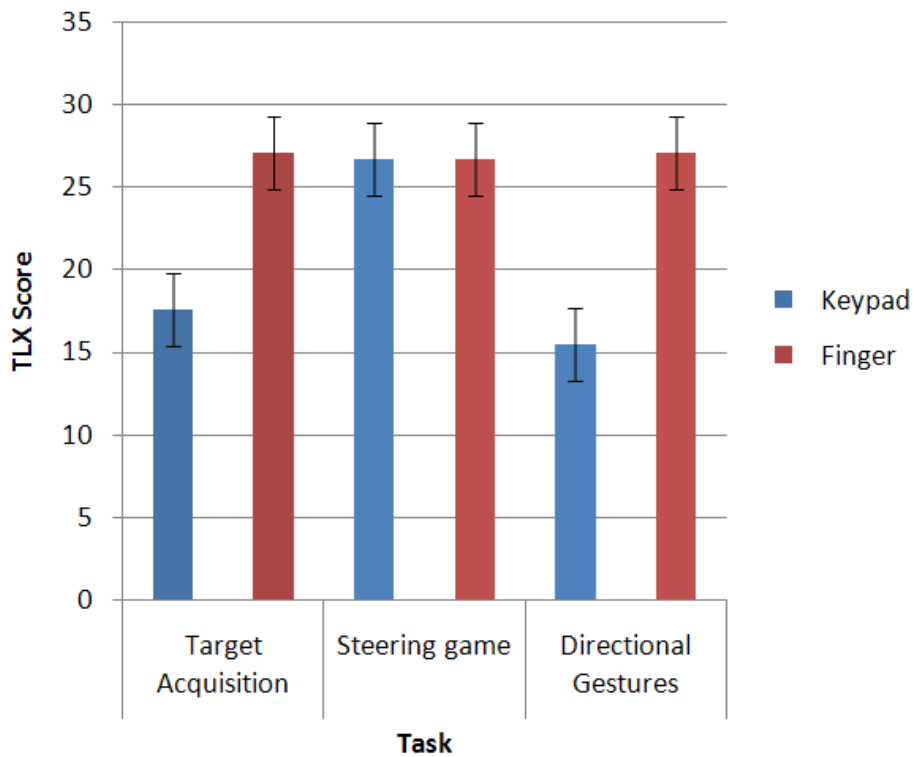


Figure 6.9: The average mean score of all three tasks

6.5.4. Preferred Interaction technique

Figure 6.10 shows the amount of users who preferred one interaction technique over the other. 58% of users preferred the keypad to the finger interaction technique in the target acquisition task. 75% of the users preferred the finger interaction technique to the keypad in the steering game. This is the only task where finger interaction was preferred by users. A large majority of users (75%) preferred the keypad to the finger interaction technique in the directional gestures task.

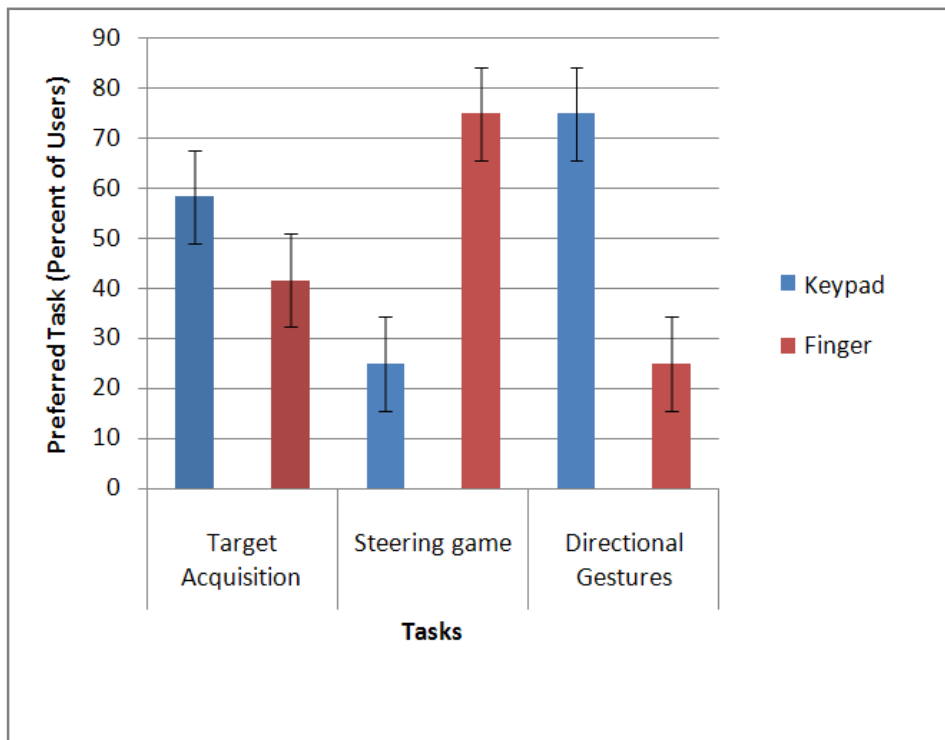


Figure 6.10: The percentage of participants who preferred one interaction technique over the other

6.5.5. Target Acquisition

The Target acquisition sub categories of the TLX questions are presented in Figure 6.11. Each category corresponds to one of the questions in the questionnaire. A paired t-test was performed to determine if there were any significant differences between the keypad and finger interaction data. No significant difference was found for mental demand ($t = 1.5, p = 0.2, df = 11$), physical demand ($t = 0.7, p = 0.5, df = 11$) and speed ($t = 0.4, p = 0.7, df = 11$). A significant difference was found for success ($t = 4.29, p = 0.002, df = 11$) (keypad: 2.9, finger: 5.4), work load ($t = 4.47, p = 0.002, df = 11$) (keypad: 3.1, finger: 5.8), and difficulty ($t = 3.86, p = 0.004, df = 11$) (keypad: 2, finger: 4.6).

Observational comments made by the observer during the user study noted that this task was the task users struggled with the most. Moving in two dimensions seemed to increase the dexterity required. This may be due to the fact that the user is simply uncomfortable with the way in which their hand must be held in front of the camera. Other tasks only required left and right movement which is easier than moving up and down as well.

The target acquisition task was scored seven points worse than the keypad on the TLX questionnaire which is a 12 % difference. For the finger interaction technique the average time on task was only 4 seconds slower, this can be accounted for by the average amount of times the users had to double back (4.25 targets

CHAPTER 6 – USER STUDY

missed per user) to select a block because they had missed it on the first attempt. The keypad exhibited a very low error score of only 0.42 targets missed per user. The t-test showed that there is a significant difference between these values. This is due to the observed difficulty users had in moving their finger in two dimensions while keeping it parallel to the camera lens. One of the users who preferred the finger interaction technique to the keypad in this task still remarked that the finger interaction technique “*was more frustrating*”. Unlike the keypad where the user could move in a straight line with very little effort, the finger interaction technique required a high level of concentration when trying to move from one target to another. This is exemplified by the significant difference between user’s mental demand when using the finger interaction technique and the keypad (see Figure 6.11). With the keypad the user simply had to keep pressing one direction button. This is exemplified by one of the users who remarked that “Motion seems faster with the finger but I kept on having a problem at a corner”. While the user could move their finger between objects at a much faster speed, actually hitting the target produced more difficulty for the user as they struggled to get the pointer to hit the targets. Fitts’ law can be applied to this drop in accuracy to try and understand why users had difficulty acquiring the targets with the finger interaction technique. Fitts’ law (Fitts, 1992) describes the time it takes to acquire a target with a rapid aimed movement. When applied to user interfaces on computers Fitts’ law can be used to describe the time it takes to acquire a button or in this case block with a pointer or cursor. Targets which are further away or smaller in size require more time to select than targets which are closer. So speed is directly proportional to accuracy. The finger interaction technique allowed users to select the blocks much faster than they could with the keypad and this led to the decrease in accuracy, as stated by Fitts’ law. Users would quickly move their finger across the screen and on many occasions miss the blocks. With the keypad technique they had much more time to line up and correct the path of the pointer with the blocks. The drop in accuracy could also be due to the sensitivity of the finger interaction technique. It could also be due to the lack of tactile feedback that users are used to when selecting something with the keypad as some users took a few seconds to register that the colour had changed and they could move on to the next target. If the amount of errors can be reduced then the time would improve and may even surpass that of the keypad.

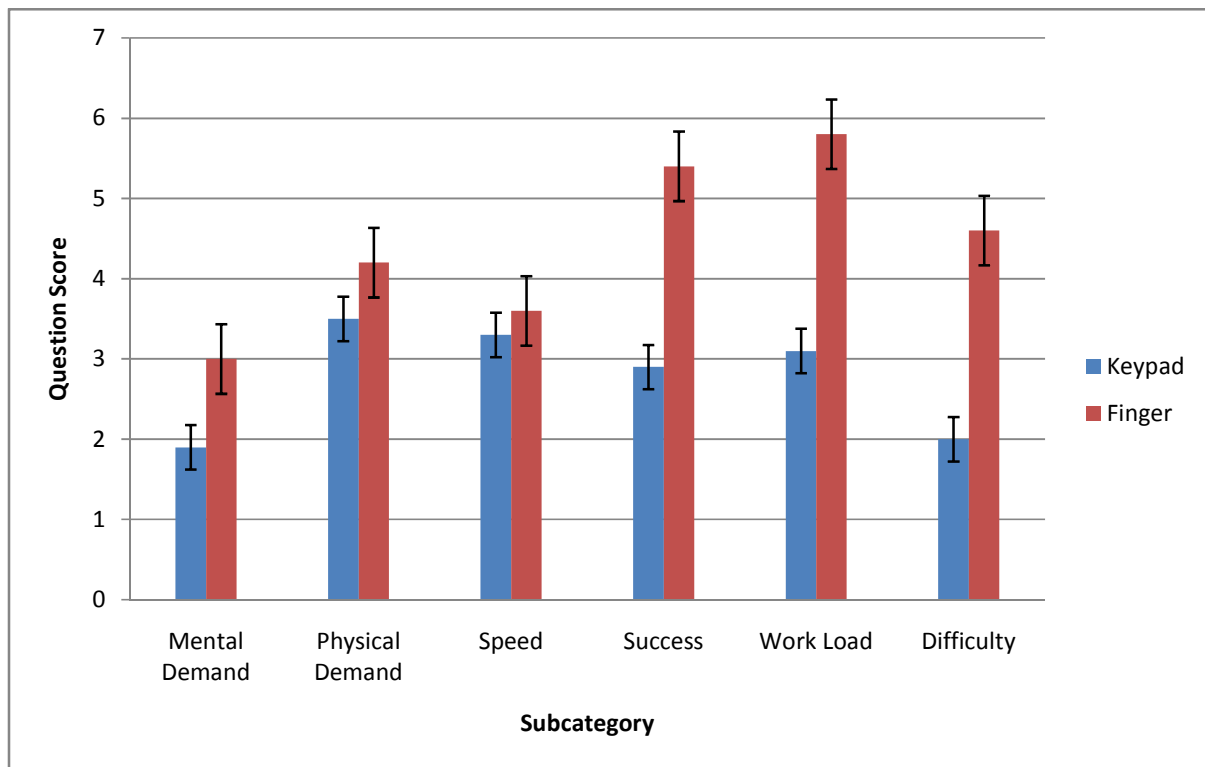


Figure 6.11: Average subcategory scores for the target acquisition task

Looking at the keypad interaction technique, users preferred it and it can be seen that the finger interaction technique performed below average with 42% users preferring it to the keypad. This however is still promising and with further work (see Chapter 8 -) mouse style interactions with the camera and finger could surpass speed and accuracy of mouse style interactions implemented using the keypad. As noted by Yamashita *et al.* (2007) moving from one type of interface to another unfamiliar one shows worse performance stats.

Many of the comments for this task expressed reserved optimism for the technique. Comments tended to say that if one thing (such as practice time, sensitivity and so forth) was changed then this technique would be better. For example one user remarked “*I think that if I had more time to learn and get used to the camera technique that it would have been easier to use*” and “*I would prefer to use the camera if it followed your finger better*”. Another user noted after they selected that they preferred the keypad for this task: “*although the camera was more entertaining*” alluding to a lack of frustration with the finger interaction technique but still preferring the keypad. Interestingly a user who selected the finger interaction technique as their preferred means of interaction on this task said that they “*Preferred camera because it is novel even though more frustrating*” commenting on the fact that even though they were faster and ultimately preferred the finger interaction technique they still experienced some frustration which was not

present with the keypad interaction. Showing that even though some users preferred the finger interaction technique they still were not completely satisfied with it.

6.5.6. Steering Game

The steering game sub categories of the TLX questions are presented in Figure 6.12. A paired t-test was performed to determine if there were any significant differences between the keypad and finger interaction data. A significant difference between the keypad and finger was found for mental demand ($t = 3.74, p = 0.11, df = 11$). No significant difference was found for physical demand ($t = 0.4, p = 0.7, df = 11$), speed ($t = 0.9, p = 0.4, df = 11$), success ($t = 0, p = 1, df = 11$), workload ($t = 1.1, p = 0.3, df = 11$) and difficulty ($t = 0.44, p = 0.6, df = 11$).

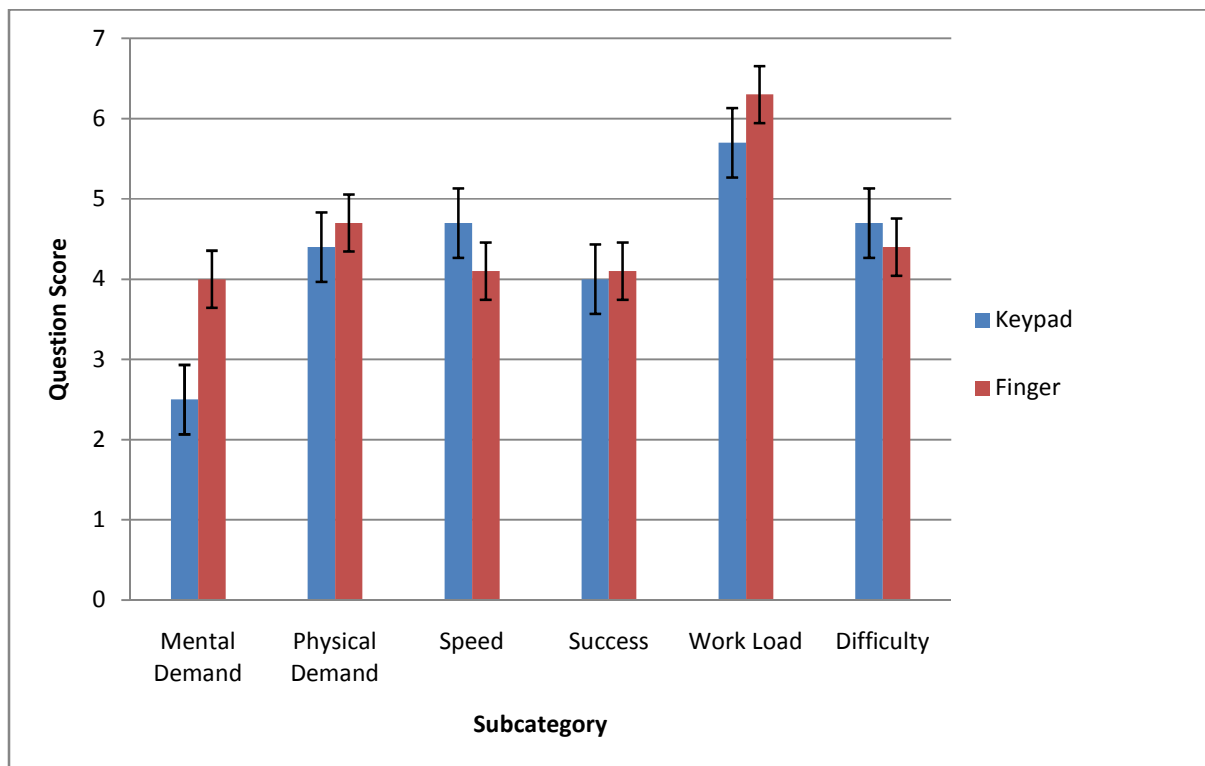


Figure 6.12: Average subcategory scores for the steering game

In the steering game the finger interaction technique received the most favorable score out of the three tasks. As will be seen this is the one task where the finger interaction technique received the most positive comments and scores. It also achieved better times and error rates. However it is important to note that participants still felt as comfortable with the keypad on average giving it the same scores as the finger interaction technique. This can be attributed to users being familiar with the keypad and not the finger interaction technique (Yamashita, Barendregt & Morten, 2007). 75% of the participants said they would rather use the finger interaction technique than the keypad. During the user study users could position the

CHAPTER 6 – USER STUDY

pointer much quicker than they could with the keypad. Users also seemed to be having much more fun with the finger interaction technique, with many continuing to play the game even after the timer had been stopped. Most of the participants said they would play the game again. Looking at the sub categories (Figure 6.12) shows that users experienced a higher mental demand and slightly higher physical demand when using their finger, this in turn leads to the finger being scored as slightly more difficult and having a higher work load than the keypad.

The time taken on this task was much faster for the finger interaction technique recording an 11.83 second gap with the keypad. It can be seen that the amount of misses is also lower for the finger interaction technique interaction technique with it scoring an average of 14.25 blocks missed and the keypad recording 17 blocks missed per user.

The reason the steering based game is more suited to the finger interaction technique is because of the analog way in which the user can control the pointer.

All the comments on the steering game were positive. Users preferred the speed with which they could catch the falling blocks. One user remarked that *“The keypad took more effort than the camera”* and another commented: *“Keypad too slow. Camera worked faster but took some time to get used to”* while another said they preferred the finger interaction technique because of a *“better scroll speed”*. Even though users had some problems getting used to the finger interaction technique they still preferred it over the well established keypad interface. Tasks well suited to analogue input such as steering games would benefit from the finger interaction technique.

6.5.7. Directional Gestures

The directional gestures sub categories of the TLX questions are presented in Figure 6.13. A paired t-test was performed to determine if there were any significant differences between the keypad and finger interaction data. No significant difference was found for mental demand ($t = 2.18$, $p = 0.06$, $df = 11$),. A significant difference between the keypad (1.8) and finger (4.3) was found for physical demand ($t = 3.3$, $p = 0.009$, $df = 11$). No significant difference ($t = 1.02$, $p = 0.3$, $df = 11$) was found for speed. A significant difference was found for success ($t = 2.37$, $p = 0.05$, $df = 11$) (keypad: 2.2, finger: 4.5), work load ($t = 2.27$, $p = 0.03$, $df = 11$) (keypad: 3.1, finger: 5.5)and difficulty ($t = 2.62$, $p = 0.03$, $df = 11$) (keypad: 1.6, finger: 4.4).

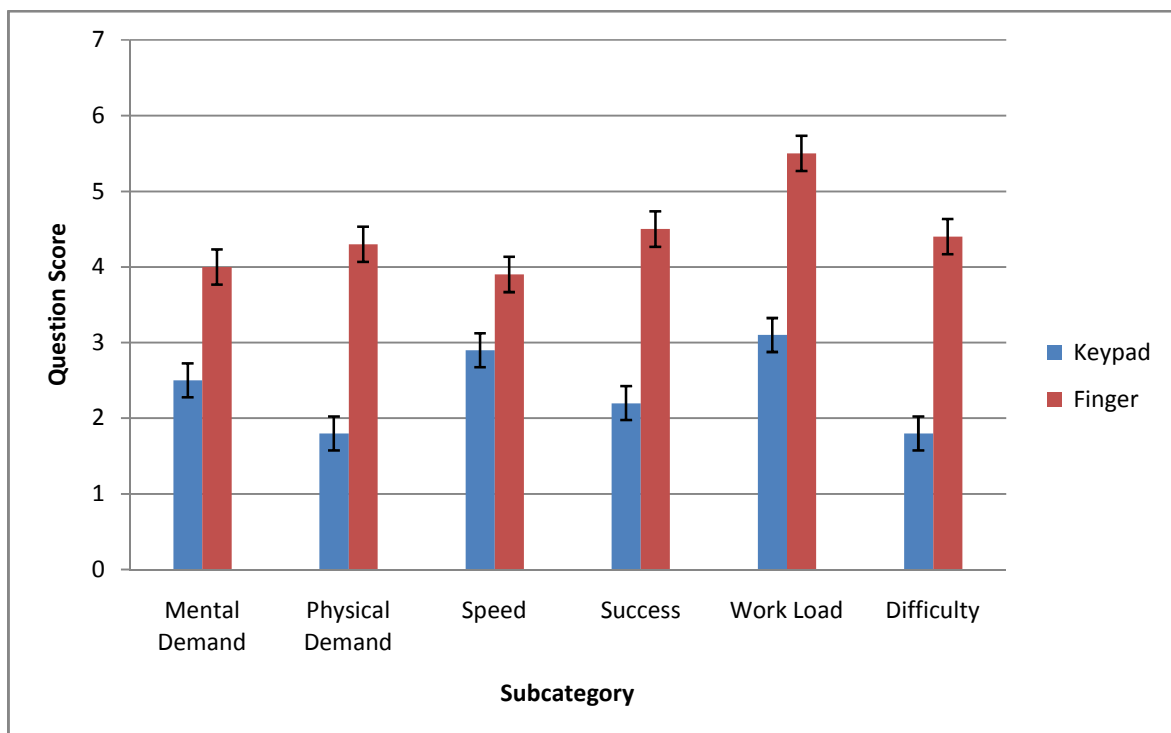


Figure 6.13: Average subcategory scores for the directional gestures task

The camera interaction technique scored badly in the TLX questionnaire. This was also the task where the keypad achieved its best score. This leads to the largest disparity between the keypad score and the finger interaction score. It was observed that the task was significantly easier to conduct with the keypad as the left and right movements of the blocks related well to the left and right buttons on the keypad. This in turn lead to users quickly completing the task taking on average 27.76 seconds as opposed to the finger interaction technique which took on average 46.03 seconds to complete.

The ease of use and speed with which the user could use the keypad in this task is reflected in the sub category scores in Figure 6.13. The finger interaction technique is significantly worse for all subcategory scores except for mental demand and speed. The work load for this task is especially high for the finger interaction technique with the score averaging at 5.5 out of 7, the highest in all tasks. All other categories which show a significant difference scored similarly with the keypad far outperforming the camera.

The user comments for this task were mostly negative. Some users complained about the sensitivity of the pointer when the finger was being tracked saying that the “*speed factor made the task longer*” and “*the pointer was slightly jumpy*”. These comments highlight the fact that this task was fairly trivial but was being slowed down and complicated by the finger interaction technique. It is important to note that simple tasks such as this one which is a straight forward selection task such as that used in a menu or picture

CHAPTER 6 – USER STUDY

gallery is better suited to an interaction technique which mirrors its simplicity (the keypad). In time as the user becomes more familiar with this camera based type of interaction they may wish to use it for these sorts of tasks exemplified by one user who commented that *“In time I might even prefer the finger interaction technique. It reminded me of early experiences with the mouse and today I prefer mouse over arrow keys”* saying that when they first started using a computer had difficulty with the mouse but now they prefer it over using the keyboard for navigation.

Three users did prefer the camera to the keypad with one saying that the technique is *“Novel and entertaining to use”*. This user also achieved good scores in all the tasks and took to the technique quickly. This is positive because the user quickly surpassed the initial difficulties of the task and could get down to focusing on the task and not how to use the interaction technique. However it must be concluded that the finger interaction technique in its current form is not well suited to directional gestures.

6.5.8. USE/SUS questionnaire

The final questionnaire (see Appendix A) is broken into four categories, namely: usefulness, ease of use, ease of learning and satisfaction. Figure 6.14 displays each of these categories as a percentage of the maximum possible score. Usefulness scores 63 % . Ease of use is slightly lower scoring 60.6% on average. Satisfaction scores in the upper sixties with 66.6%. The highest scoring category of the interaction technique is ease of learning, with users scoring it at 75.3%.

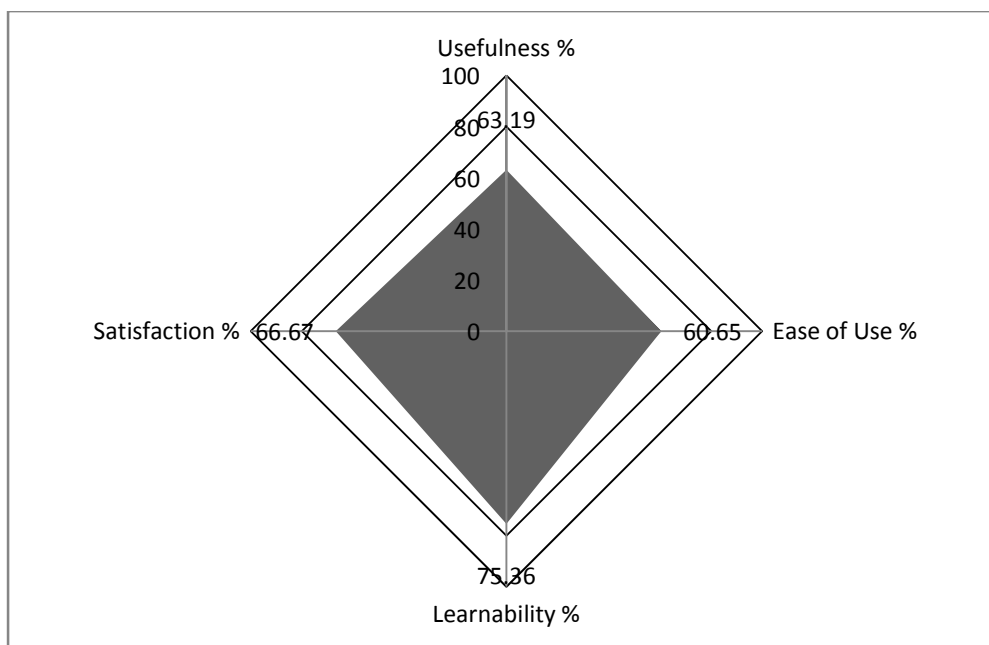


Figure 6.14: Usefulness, ease of use, ease of learning and satisfaction for participants using the camera interaction technique. Higher is better

CHAPTER 6 – USER STUDY

Usefulness, ease of use and satisfaction all score similarly in the low to mid sixties. Ease of learning scores the highest of the four, at 75.36%. At first this is surprising as many of the users initially had some trouble with the finger interaction technique, but as this questionnaire was given to the user after all the tasks were completed, they had had ample time to learn and understand how to fully use the technique, hence the high score. The next highest scoring category was satisfaction, with users scoring this category at 66.6%. This means that users found the satisfaction of the system above average, agreeing in part that they enjoyed using it and would recommend it to others. For tasks such as games this category would need to score quite high in order to be effective as the user needs a high level of satisfaction with the interaction technique to fully enjoy the game. A game or steering based application should not make the user concentrate on the interaction technique to the detriment of the actual application's content (Wisniewski, et al., 2005). This would mean that ease of use which scored 60.6% would also need to be improved for these sorts of tasks.

6.6. Summary

This chapter has presented a user study which was conducted to ascertain the user experience through the use of three tasks based upon Foley's. The first task was the target acquisition task which assessed the user's ability to interact with the mobile phone using their finger as a mouse style pointer. The finger interaction technique scored slightly below the keypad in all performance metrics but still showed promise with regards to future work. The second task was the steering game. This task was the only task where the finger interaction technique outperformed the keypad in some of the performance metrics. Users gave favorable comments after playing the game and 75% of them preferred the finger interaction technique to the keypad. The final task was the directional gestures task which asked users to navigate three coloured blocks either left or right with the ability to select the desired one. The keypad far outperformed the finger interaction technique because of the simplicity of the task which was well suited to simple left and right button presses and not the gestures utilized by the finger interaction technique. This leads to the conclusion that simple tasks should utilize the simplest interaction techniques, in this case the keypad as the finger interaction technique presents too much overhead to the user.

The final part of the user study presented the participants with the USE/SUS questionnaire made up of four categories. These categories contained questions which dealt with usefulness, ease of use, satisfaction and ease of learning. Users scored these questions on a 7 point Likert scale. The results of this questionnaire revealed that ease of learning was the highest scoring, followed by user satisfaction and usefulness. Ease of use scored the lowest. These results showed that once the user had used completed the user study they felt comfortable and satisfied enough to score ease of learning and satisfaction as the highest.

CHAPTER 6 – USER STUDY

The final conclusion of the user study is that steering based tasks and games are well suited to the finger interaction technique. The keypad interaction technique is well entrenched in mobile phone users and is well suited to simple tasks and navigation like the directional gestures task. The data gathered from the target acquisition task shows promise but more users preferred the keypad.

Chapter 7 - Discussion

This chapter presents a discussion of the camera based finger interaction technique in relation to the initial problems described in Chapter 1. A look at the way in which the finger interaction technique relates to other implementations in the mobile computer vision field is also presented. The results obtained from the user study are examined in terms of the overall usability of the finger interaction technique. Finally a discussion on mobile applications and games in relationship to the finger interaction technique is presented.

7.1. Discussion

This study aimed to investigate the use of the mobile phone camera as an interface for interactive applications, most notably as a substitute to the keypad in steering based tasks and mouse based interactions. The main reason for this interaction technique was to alleviate the restrictions imposed by the keypad interface currently found on mobile phones. Occlusion when using touchscreens was also a problem this study aimed to solve. Touch screens can be found on some high end mobile devices but they are limited to these high end handsets. However, cameras can be found on the majority of mobile phones and present the potential of touchscreen style gestures and interactions without the problem of occlusion.

Many mobile computer vision studies were explored in Chapter 2. Across all these studies it was visible that mobile computer vision was possible and could be deployed to mobile phones. Back-of-device interactions were shown to not obscure the small screen present on mobile phones (Baudisch & Chu, 2009). Various optical flow and motion detection algorithms were examined and a thresholding algorithm was selected with the intent of deploying it onto a Nokia N95 smart phone. The algorithm used no pre processing and implemented a search square. This search square allowed increased frame rate without a decrease in accuracy. The lack of pre processing was advantageous in the increased framerate it allowed. This enabled real time feedback to be given to the user which is important in computer vision systems (Moeslund & Granum. 2001). The main drawback of the multiband thresholding algorithm was its inability to distinguish between colours if they were within the same threshold, regardless of whether they were part of the finger or not. This would lead to low accuracy scores if the finger is the same colour as the background which are most notably dark lighting conditions or finger colour backgrounds. So the finger interaction technique would work in many offices and homes with adequate lighting but could not compare to a backlit keypad in the dark, this limited the amount of locations it can be used in.

CHAPTER 7 – DISCUSSION

Moeslund and Granum (2001) categorise computer vision application: surveillance, control and analysis. The finger interaction technique described in this theses falls within the control category and the primary performance requirements (and their ranking) of a control system are from least important to most important: robustness (0 or 1), accuracy (1) and speed (2). For this study it was decided that robustness was an important consideration because of the portable nature of mobile phones and the different environments they are exposed to. Hence robustness was ranked 2nd along with accuracy.

Chapter 4 addressed the issue of speed within a control system and because it is the highest ranked most important problem a series of benchmarks were performed to identify the fastest API. It was found that C++ yielded the fastest results for all tests with a framerate (10 FPS) above the minimum frame rate suggested in the literature of at least 6 FPS (Beier, et al., 2003). The main disadvantage of using C++ is that it is limited to a small subset of handsets, in this case those with Symbian S60 installed. It seems that there is still a need for a software interface which can be deployed onto the majority of handsets with the capability of providing real time tracking. Java shows promise but is by no means ready to provide widespread real time camera tracking interactions.

It was unfortunate that Java did not provide a way to access the viewfinder stream like Python (using a wrapper to NokiaCV) and C++ do, as Java showed promising scores throughout the experiment and has the highest deployability across mobile phone platforms.

Chapter 5 addressed the accuracy of the system and it was shown that the finger interaction technique was not limited to a single lighting condition and background but could be used in the portable mobile environment and could adapt to differing conditions. From this it was shown that the finger interaction system was robust enough to be used in a mobile environment. Hence the finger interaction technique fulfilled the three main performance requirements of a control computer vision system (Moeslund and Granum, 2001). The system worked well in adequately lit environments. A user study was then conducted to ascertain user satisfaction and usability of the finger interaction technique in relation to the standard keypad.

The user study presented in Chapter 6 illustrated user's satisfaction with regard to the types of interaction techniques. To summarize the user study findings users were tentative in accepting the finger interaction technique for mouse based interactions with the finger interaction technique receiving slightly lower scores than keypad. Users did not like the directional gesture style interactions when using the finger interaction technique and ranked it significantly worse than the keypad. However users were satisfied with the analogue steering based input afforded to them by the finger interaction technique. This was exemplified in

CHAPTER 7 – DISCUSSION

the higher scores and better comments attracted by the steering based task. These simple steering based tasks can mostly be found in games (Wisniewski, et al., 2005) and fall into the position section of subtasks described by Foley *et al.* (1990). It would be beneficial to revisit Foley's taxonomy to examine where the finger interaction technique can be improved and where it would be better to use a keypad.

Each of Foley *et al.*'s (1990) subtasks specify the building blocks of an interactive system. Each task in the user study was designed to examine a selection or combination of subtasks from the taxonomy. Quantify, text entry and orient were not examined as the finger interaction technique is not well suited to these tasks. Target acquisition encompassed position and pathing in two dimensions. This task focused on mouse based interactions. The steering game encompassed position in one dimension, focusing particularly on steering based interaction. The directional gestures task encompassed pathing, selection and gestures in one dimension. This task focused on using gestures with the finger interaction technique which is an addition to the Foley taxonomy by Reimann and Paelke (2006) discussed in Section 2.3.3.

The performance and usability data gathered from the users in each of the user study tasks can be applied to the taxonomy subtasks within each of the user study tasks. Table 7.1 shows an overview of each of the subtasks and how they relate to the three types of interactions. Scores are reduced to the two point system used by Moeslund and Granum (2001) by the way in which they were ranked in the user study with regard to the finger interaction technique. The Steering game was ranked first in the user study and so the subtask position receives a 2 (most important). The target acquisition task was ranked second in the user study and so pathing receives a 1 (second most important). The directional gestures task was ranked last and so the gestures subtask which is unique to this task receives a 0 (third most important).

As can be seen in the last three columns of Table 7.1 each interaction technique encompasses the subtasks which it uses. Analogue Steering Interactions which include position are the most favoured by the finger interaction technique. Mouse Style Interactions include the two high scoring subtasks but pathing is shared with the Gestures and Simple Navigation interaction technique which was ranked last and hence a lower score of 1 is achieved. The final interaction technique of Gestures and Simple Navigation includes all subtasks but position. The two subtasks which are unique to this interaction technique are both given 0. This leads to the conclusion that simple gestures and navigational inputs are not well suited to the camera based interaction technique. The Position subtask is well suited to the finger interaction technique, whereas pathing shows promise. Hence the finger interaction technique is best suited to analogue steering interactions which encompass the position subtask.

CHAPTER 7 – DISCUSSION

Table 7.1: Five sub tasks are listed in the first column. The first four are from Foley *et al's* original taxonomy. Gestures is an addition to the taxonomy by Reimann and Paelke. The ratings are from user's ratings, data and feedback in the user study and are reduced to fit within the 2 point system shown in this table.

	Overall Rating	Interaction Techniques		
Position	2	Analogue Interactions	Steering	Mouse Style Interactions
Pathing	1			Gestures and Simple Navigation
Select	0			
Gestures	0			
Overall rating:		2	1	0

The following section relates back to the topic of multimodal interfaces and readdresses whether the finger interaction technique accomplished the goals of multimodal systems set using TYCOON.

7.1.1. Finger Interaction as a Multimodal Interface

Section 2.2 presented the various advantages of using multimodal input to better the goals of computer interfaces. Using TYCOON (TYpes and goals of COOperation) (Martin, 1998) resulted in the classification of the finger interaction technique as an equivalent or alternate type of interaction. This type of interaction allows for faster interactions and better recognition by the user. The previous section showed that the finger interaction technique is best suited to analogue steering interactions, hence only this type of interaction is addressed for the remainder of the chapter.

7.1.1.1. Faster Interactions

In order to ascertain whether the finger interaction technique allows the user to interact faster with the mobile phone, the results of the user study must be reexamined. In the three tasks in the user study no statistically significant difference was found between the times it took to complete the tasks on the camera to the time it took on the keypad. However the mean time of the finger interaction technique in the steering game was 12 seconds faster. This shows that in the steering game users on average were faster when using the finger interaction technique than when using the keypad. It is also important to note that users have had many years of experience with the keypad and had only a few minutes of experience with the camera. Hence it can be said that for steering based interactions such as those exhibited in the steering game the finger interaction technique does allow for faster interactions.

CHAPTER 7 – DISCUSSION

7.1.1.2. Recognition

Recognition refers to the ability of the multimodal system to recognize commands from two or more modalities. It was shown in the user study results that the keypad is well suited to many tasks on the mobile phone. Simple gestures for instance are easier to perform with a key press than by trying to move ones finger in small motions in front of the camera. In an instance such as this the user could use the keypad to browse a menu or list of objects and then move to the finger interaction technique if they then need to input analogue steering based information. This increase the accuracy of the recognition of commands for the system as it will expect key presses for simple tasks and camera input for analogue ones not suited to the digital keypad buttons. Therefore the finger interaction technique can be said to aid in the recognition in a system and satisfies this goal of equivalence.

Both of the goals of equivalence for steering based interactions were satisfied by the finger interaction technique. Another goal of any portable system is to increase the amount of time the device can stay operational i.e. battery life. The amount of power drain on the mobile phone is investigated in the following section.

7.1.2. Battery Life

One concern with the finger interaction technique is the continuous use of the camera and its drain on battery power. Battery lifetime is an important consideration for new interaction techniques as users will respond negatively if there is a substantial decrease in battery lifetime due to the interaction technique (Rahmati & Zhong, 2008). A study was conducted by Wang *et al.* (2006) using their camera interaction technique. Three tests were run namely: Only keypad active, camera with little interaction and no backlight and camera with exhaustive usage and no power saving features enabled. Their study showed a significant drop (from 8 hours to 3 hours on a Motorola v710 mobile phone) in running time when the camera interaction technique was used.

It is not surprising that the battery life is affected by constant tracking of the camera stream. The highest drain on current phones is the backlight and constant radio use (Wang, et al., 2006). Hence as it is known that these features drain battery power various power saving features have been enabled to extend battery life, such as switching off the backlight after a few seconds of no operation. If the finger interaction technique were to be widely deployed there would have to be the same type of power saving features deployed as there are for current hardware solutions. This would allow the finger interaction technique to be widely deployed to many systems and used in many applications. These applications are discussed in the following section.

7.1.3. Deploying the Finger Interaction Technique

The previous sections have shown that the finger interaction technique can be run on current mobile phone technology and is usable for certain interaction techniques. This section provides a discussion on the various games and applications which could be well suited to the finger interaction technique.

As mobile technology has grown, so too has the demand for more interactive based activities on mobile devices. This can be seen by the huge growth in the mobile gaming sector (Wisniewski, et al., 2005). Mobile games are however, limited by both hardware performance and the physical size and placing of the keypad. Large scale games like those seen on consoles and PCs are not possible to implement on such physically small hardware.

Mobile games would benefit from an analogue interface such as the finger interaction technique. The main disadvantage with the current way of analogue interfacing with games is that the user must press keys multiple times to try and emulate analogue input. This technique of interfacing with games is prominent in arcade games where the user is required to either ‘button bash’ to achieve a result or to press a series of buttons in the correct order at a certain speed to attain the required goal. It is however unlikely that these games work well on mobile devices as the keys on phones are not designed to be pressed multiple times in succession at speed. If an analogue interface is required but the hardware is not available, then certain solutions can be implemented, such as holding down a button to increase a value (e.g. hold down up to increase acceleration at a constant rate). This however is not very efficient when precise values are required at very precise intervals.

The games that are best suited to analogue interfacing are any games that require fluid movement such as first person shooters or racing games. A genre of games which is well suited to the finger interaction technique is that of racing games as these exemplify the analogue steering interactions discussed in the previous section. It should be noted that the majority of racing games require analogue movement in only one dimensions with relation to the player i.e. the player is only presented with controls to move left and right with forward and backward mapping well to the keypad. Currently key presses are used to initiate slides and other racing maneuvers which are sufficient to successfully control the vehicle through any given obstacles. However, realistic type games take into account real world physics and the corresponding action of simply pressing a button (turn steering wheel 100% right or left) would result in failure in more realistic type of game. This is why a steering wheel or analogue gamepad is required for realistic racing simulations on consoles and PCs. While a keypad can provide a good interface to a well designed game in two

CHAPTER 7 – DISCUSSION

dimensions it is however very limiting when moving to the third dimension (up and down). It can clearly be seen that there are very few if no realistic type games which are deployed on mobile devices.

As stated by Heumer *et al.* (2006) there are various types of advanced games technology which are coming to the fore such as the *Eyeto* used in the *Sony Playstation* console series which captures a gamers movements and project Natal on the Xbox 360 which tracks the user's body as a controller. Hardware solutions such as the Nokia Xpress-on gaming cover provide the user with an extended keypad which makes interfacing with mobile games easier (Nokia, 2002). This however, is still an added extra which one must purchase to add to the gaming experience. Users would rather buy a dedicated mobile gaming platform like the Sony PSP (Playstation Portable) or Nintendo DS than buy specialized hardware for their Mobile Phone. Mobile devices lend themselves better to casual gaming than the more demanding games available on specialized gaming systems.

Currently the mobile gaming market is dominated by arcade and puzzle style games (Wisniewski, et al., 2005). This is because of the keypad and the convenience of the casual game. This is because of the amount of concentration required when playing games on the move. Games which are ported from other platforms to mobile devices and which bear the same name (e.g. *Need for Speed*, *Grand Theft Auto*, *Call of Duty*, *Fifa*, *Halo*) are almost always changed in genre to suit the interface and environment (short gameplay on the move) provided by mobile phones. Realistic versions of games such as first person shooters are not suited to the current mobile phone environment as they require extensive multi modal input to provide the player with a realistic interface.

Any recently released mobile phone has the hardware to play mobile games. Many of these consumers are not going to spend extra money on such merchandise as the Xpress-on gaming cover or other types of peripheral hardware. Using the hardware already available as a fluid gaming interface is important for the user to be able to enjoy mobile gaming without investing large sums of money into it.

Games which require pointing devices, a large screen or convenience of no additional hardware will not work well at all on the mobile platform (Nokia, 2003). The finger interaction technique with further work could allow for such games (tailored for a small screen) to work on mobile platforms.

Another category of programs which could benefit from the interaction technique is that of mobile applications. These applications form an integral part of the appeal of mobile devices. The advent of fully fledged internet and QWERTY keyboards has allowed phones to perform almost all of the functions that a mobile computer can perform being only limited by their size and processing power. Mobile applications range from word processors to web browsers. All of these applications are also limited to the layout of a

CHAPTER 7 – DISCUSSION

mobile device's interface. While QWERTY keyboards have allowed much more flexible character input mechanisms they are still have the disadvantages of being small and tend to increase the size of the device. The target acquisition task map well to web browsers, with further work the finger interaction technique could also be used for browsing the internet or large documents. As noted the finger interaction technique and the keypad interact through equivalence so users could move a pointer around with their finger and then conduct simple tasks like turning pages with the keypad if they so wish. Mobile applications can benefit greatly from a multimodal interaction technique like the finger based interaction.

7.2. Summary

This chapter has discussed the finger interaction technique in relation to the literature supporting it, the algorithms chosen from said literature and the various tests which were performed on it. The accuracy and speed tests showed that the required levels of robustness, accuracy and speed were met. Finally the user study results were analyzed.

The finger interaction technique yielded good results when used in steering based tasks. Applications which use simple steering based interactions will benefit from the finger interaction technique. Speed benchmarks provided the best platform at current technology levels at the time of the study. Various accuracy tests identified the lighting conditions and backgrounds which work best with the technique. Finally the user study solidified the initial claim that the finger interaction technique would help steering based actions and games on mobile phones.

As such the finger interaction technique has met one of its goals of increasing user satisfaction and usability in steering based tasks. The goal of implementing gestures with the finger interaction technique was not met. Users found the keypad to yield more satisfying results and as such found the keypad more usable than the finger interaction technique for simple directional gestural input. It is unclear whether the goal of implementing mouse based interactions was met. According to the data gathered in the user study the technique performed at the same level as the keypad for users in mouse based interactions and target acquisition. This is encouraging as users were only exposed to the finger interaction technique for a few minutes as opposed to the years of experience with keypad interactions they have had. However, because the finger interaction technique did not show a significant difference over the keypad it cannot be said that the goal of implementing mouse based interactions has been met. With more practice and some additions mentioned in the following chapter users could find the technique more satisfying for mouse based interactions, encompassing the majority of navigation and analogue interaction tasks performed on mobile phones.

Chapter 8 - Conclusion and Future Work

This chapter presents a conclusion to the entire document. Focusing on the four main areas of the main problem this study addressed. These are: the viability of deploying a computer vision system on mobile phones, what type of algorithm to deploy and what software interfaces exist to accomplish this, how the accuracy and robustness of such a system would be affected by changing lighting and background conditions and whether it is a usable and satisfactory interaction technique to users of mobile phones. This is followed by a discussion of the future work which could lead to a more usable interaction technique.

The high level of mobile phone penetration and the fast growth of powerful processing potential, as well as problems with current interaction techniques such as occlusion on touchscreens and lack of analogue input on keypads were motivators to explore the deployment of a finger based interaction technique on mobile phones. Drawing on the current literature of computer vision and vision based systems a finger interaction technique was designed to be practically deployed to the N95 mobile phone running the S60 operating system. It was found that there was a need for these computer vision systems and it was possible to deploy them onto mobile phones. A multiband thresholding algorithm was selected because it did not require pre processing and scaled well to the pixel stream received from the camera. This solved the first problem of identifying the viability of deploying a computer vision system on mobile phones.

The multiband thresholding algorithm was then tested on three APIs and C++ was identified as the most viable to implement the finger interaction technique. The Java and Python APIs did not perform fast enough to allow real time interactions. This solved the second problem area of selecting a suitable algorithm and identifying the effectiveness of current mobile phone software interfaces for real time visual interactions.

As well as speed the other important factors in a computer vision system are robustness and accuracy. These were explored through a series of accuracy tests. These tests showed that the finger tracking system was accurate and robust enough to be used in many adequately lit environments, solving the third problem of whether the finger interaction technique could be used in various differing conditions.

Finally a user study was conducted to explore the user satisfaction and usability of the finger interaction technique. The three separate tasks examined in the user study showed where the strengths and weaknesses of the technique lie. It was discovered that the system is well suited to analogue steering based interactions. Task times and amount of errors were better than those attained by users with the keypad. Comments were

CHAPTER 8 – CONCLUSION AND FUTURE WORK

also positive for steering based interactions and the majority of users preferred the finger interaction technique to the keypad which they had been using for much longer than the finger. The finger interaction technique shows promise for target acquisition and mouse based interactions but is not suited to gestural and simple navigational interactions, these would be better dealt with by the keypad in a multimodal system. This leads to the conclusion that finger based interactions using the camera on the mobile phone are well suited to steering based interactions, show promise for mouse based interactions and are currently not suited to directional gestures and simple navigation. This solved the final problem of whether users were satisfied with the finger interaction technique and whether it was usable by identifying the areas in which it is.

It was shown that there is a need for new types of multimodal interaction techniques on mobile phones. Steering based interactions were shown to benefit from the finger interaction technique without the problem of occlusion or needing to increase the size of the device. The keypad on phones is suited to some tasks such as simple navigation and combining the finger interaction technique with the keypad would create a powerful multimodal interface. The ways in which these interaction techniques cooperate have by no means been fully explored and there is much future work which must still be done to fully expose the advantages of vision based interaction techniques on mobile phones.

8.1. Future Work

During the user study a few issues with the interaction technique arose. Firstly the high sensitivity of the technique caused difficulty for some users as they said the technique was highly susceptible to small movements. This was especially noticeable in some people who had shaky hands. Secondly the jumpiness of the pointer affected some of the users. The pointer would sometimes make small jumps to different parts of the finger. This did not affect large sweeping motions of the finger but became an annoyance when it sometimes occurred when users were making small fine movements. This chapter presents approaches to solving these problems as well as some possible extensions to the finger tracking system.

The two issues users had the most problems with was the high level of sensitivity and the jumpiness of the pointer when trying to carry out small movement tasks. The first issue of sensitivity is closely tied to how the user moves their finger. The system draws the pointer where it detects the tip of the finger. It may be that the users were simply not used to working with finger interactions that they felt it moved too fast. This is increased by the fact that the user is receiving no resistance to their finger as the surface of a touchscreen would provide. To try and negate this high sensitivity a grid could be used to draw the pointer. The grid would be updated when the finger is detected in one of the blocks and the block highlighted as in Figure

CHAPTER 8 – CONCLUSION AND FUTURE WORK

8.1. This would mean that the user must move their finger a much larger distance for the pointer to be updated.

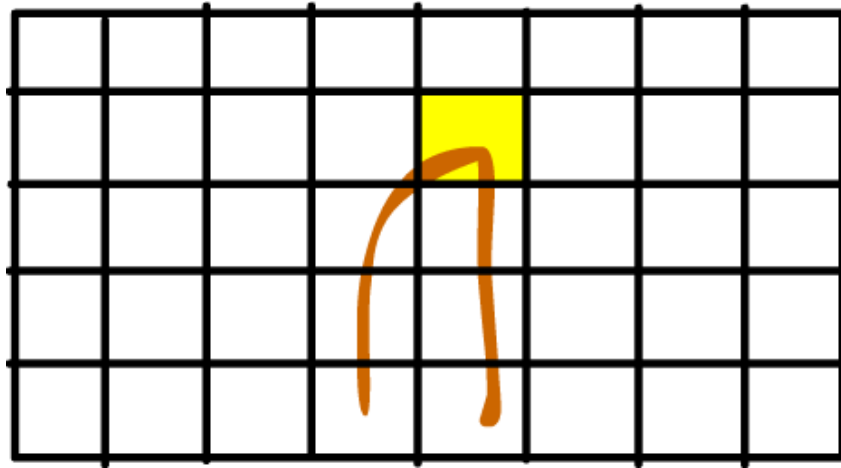


Figure 8.1: The screen is divided into a grid. The top of the finger is detected in the shaded block, this is also the pointer. As long as the finger stays in this block the pointer will not move. As soon as the finger moves to another block the pointer will then be drawn there.

This would solve both the sensitivity complaint and the jumpiness issue. However this will decrease the accuracy as very small movements will not be registered unless the finger moves from one block to another. This would address both the problem of sensitivity and the jumpiness experienced by users but at the expense of small precise actions.

In Section 5.3 it was shown that using the system in an environment where the colours change such as on the move or under moving shadow causes a drop in accuracy. To help alleviate this problem and extend the range of environments the colour being tracked could be changed from frame to frame. As there is a small leeway granted by the thresholding algorithm the reference colour could be updated as it changes with the environment. This would allow the system to adapt to the changing environment and hence increase the accuracy. However there may be a problem with accuracy if the algorithm detects the wrong colour. Further research is required into the accuracy of the algorithm in dynamic lighting conditions.

While speed was not an issue, as noted in Chapter 4 -the scope with which the system can be deployed is limited to the Symbian S60 operating system. A wrapper class for Java could be created which has access to the bitmap stream from the camera. This would allow the frames per second to hopefully reach an acceptable level in Java. Coupled with the reasonable scores achieved in the other speed benchmarks the system could be deployed to many more mobile devices than just the Nokia phones which have Symbian S60 installed on them.

CHAPTER 8 – CONCLUSION AND FUTURE WORK

Currently to use the system an application must be coded using the finger tracking classes. It would be beneficial if instead of having to program a new application every time the finger interaction technique was required, the system could be set up to issue Symbian commands such as `left` and `right`. This would allow any application to be used with the finger tracking system such as internet browsers and mobile games.

As noted in the previous chapter battery life is important. It would be beneficial to conduct an experiment on how the finger interaction technique's use of the camera affects battery life.

References

- Akenine-Moller, T. and Ström, J.** (2003). *Graphics for the masses: a hardware rasterization architecture for mobile phones*. International Conference on Computer Graphics and Interaction Techniques. San Diego. Pages 801 - 808.
- Arribas, P., Macia, F. and Monasteria, H.** (2000). *FPGA Implementation of Camus Correlation Optical Flow Algorithm for Real Time Images*. Dpto. de Sistemas Electrò y de Control. E.U.I.T Telecomunicaciòn. Univ. Politècnica de Madrid, Madrid.
- Ballagas, R., Rohs, M. and Sheridan, J.** (2005). *Sweep and Point & Shoot: Phonecam-Based Interactions for Large Public Displays*. Computer Human Interactions. Portland, Oregon.
- Barron, J., Fleet, D. and Beauchemin, S.** (1994). *Performance of Optical Flow Techniques*. International Journal of Computer Vision. pp. 43-77.
- Baudisch, P. and Chu, G.** (2009). *Back-of-Device Interaction Allows Creating Very Small Touch Devices*. CHI 2009 ~ New Mobile Interactions, Boston, MA.
- Beier, B., Stichling, B. and Kleinjohann, B.** (2003). *Marker-less Vision Based Tracking for Mobile Augmented Reality*. Technische Universit"at Ilmenau, Universit"at Paderborn.
- Bhandari, S. and Lim, Y.** (2008). *Exploring Gestural Mode of Interaction with Mobile Phone*. CHI Proceedings, Florence.
- Bretzner, L., Laptev, I. and Lindeberg, T.** (2002). *Hand Gesture Recognition using Multi-Scale Colour Features, Hierachical Models and Particle Filtering*.
- Brooke, J.** (1996). *SUS - A Quick and Dirty Usability Scale*. Usability Evaluation in Industry. Taylor and Francis.
- Bucolo, S. and Billingham, M.** (2007). *User experiences with mobile phone camera game interfaces*. Proceedings of the 4th International Conference on Mobile and Ubiquitous Multimedia. Christchurch. ACM Press. pp. 87 - 94.
- Cairns, P. and Anna, L.** (2008). *Research Methods for Human-Computer Interaction*. Cambridge University Press, 2008. Cambridge.
- Capin, T. Haro, A. Setlur, V. and Wilkinson, S.** (2006). *Camera-based Virtual Environment Interaction on Mobile Devices*. Lecture Notes in Computer Science.

APPENDICES

- De Castro, E. and Morandi, C.** (1987). *Registration of Translated and Rotated Images Using Finite Fourier Transforms* . IEEE Transactions on pattern analysis and machine intelligence.
- De Micheli, D., Torre, V. and Uras, S.** (1993). *The Accuracy of the Computation of Optical Flow and the Recovery of Motion Parameters* . IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 15, No. 5.
- Dhome, M., Richetin, M., Lapreste, J. and Rives, G.** (1989). *Determination of the Attitude of 3D Objects from a Single Perspective View*. IEEE Transactions on Pattern Analysis and Machine Intelligence. pp. 1265 - 1278.
- Drab, S. and Artner, N.** (2005). *Motion Detection as Interaction Technique for games & Applications on Mobile Devices*. Upper Austria University of Applied Sciences. Hagenberg.
- Engineering Toolbox, The.** (2005). *Illuminance - Recommended Light Levels*. [Online]. Available at: http://www.engineeringtoolbox.com/light-level-rooms-d_708.html (Accessed April 2008).
- Faulkner L.** (2003). *Beyond the five-user assumption: Benefits of increased sample sizes in usability testing*. Behavior Research Methods, Instruments, & Computers. 35(3), 379-383.
- Fitts, P. M.** (1992). *The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement*. Journal of Experimental Psychology, 47(6): pp: 381 - 391, 1954. (Reprinted in Journal of Experimental Psychology: General.
- Foley J., van Dam, A., Feiner, S. and Hughes, J.** (1996). *Computer Graphics - Principles and Practice, Second Edition in C*. Addison Wesley.
- Foley, J., Wallace, V. and Chan, P.** (1990). *The human factors of computer graphics interaction techniques*. Human-computer interactions, Prentice Hall Press. pp 67 - 121.
- Forman, G. and Zahorjan, J.** (1994). *The Challenges of Mobile Computing* . IEEE Computer 27 (4).
- Freeman, T., Tanaka, K., Ohta, J., and Kyuma, K.** (1996). *Computer Vision for Computer Games* . Mitsubishi Electric Research Lab, Killington.
- Futuremark.** (2009). *SPMark*. [Online]. Available at: <http://www.futuremark.com/products/spmarksymbianos9/> (May 31, 2009).
- Geiger, C., Kleinnjohann, B., Reimann, C. and Stichling, D.** (2001). *Mobile AR4ALL* . Proceedings of the IEEE and ACM International Symposium on Augmented Reality.

APPENDICES

- GSMarena.** (2009 a). *Nokia N95 Full Specification*, GSMarena. [Online]. Available at: http://www.gsmarena.com/nokia_n95-1716.php. (Accessed May 08, 2009).
- GSMarena.** (2009 b). *Toshiba TG01*. [Online]. Available at: http://www.gsmarena.com/toshiba_tg01-2662.php (Accessed August 2009).
- Guan, Poh.** (2003). *Developing MIDP 2.0 Applications For Nokia Devices*. [Online]. Available at: <http://www.forum.nokia.com>. (Accessed February 2009).
- Hanlon, M.** (2007). *The Tipping Point: one in two humans now carries a mobile phone* [Online]. Available at: www.gizmag.com/mobile-phone-penetration/8831/ (Accessed December 2008).
- Hannuksela, J., Huttunen, S., Sangi, P. and Janne, H.** (2006). *Motion-Based Finger Tracking for User Interaction with Mobile Devices*. IEEE Transactions on Multimedia, 8(5): 956 - 972.
- Haro, A., Mori, K., Setlur, V. and Capin, T.** (2005). *Mobile Camera-based Adaptive Viewing*. - Christchurch, New Zealand : 4th International Conference on Mobile Ubiquitous Multimedia.
- Haro, A., Mori, K., Capin, T. and Stephan, W.** (2005). *Mobile Camera-based User Interaction*. IEEE International Conference on Computer Vision Workshop on Human-Computer Interaction, Beijing, China.
- Harper, R.** (2003). *People Versus Information: the Evolution of Mobile Technology*. Proceedings of the 5th International Symposium on Human-Computer Interaction with Mobile Devices and Services. (Mobile HCI '03). Udine, Italy. pp 1 - 14.
- Hart, S. and Staveland, L.** (1988). *Development of a multi-dimensional workload rating scale: Results of empirical and theoretical research*, In P.A Hancock & N Meshkati, Human Menatal Workload. pp. 139-183 . Amsterdam, The Netherlands. Elsevier.
- Hauptmann, A.** (1989). *Speech and gestures for graphic image manipulation*. Proceedings of the Conference on Human Factors in Computing Systems. New York. Vol 1, ACM Press.
- Heumer, G., Carlson, D., Sree H., Maheshwari, S., Hasan, W., Jung, B. and Scharader, A.** (2006). *Paranoia Syndrome - A Pervasive Multiplayer Game using PDAs, RFID and Tangible Objects*. Virtual Reality and Multimedia Group, ISNM, Lübeck.
- Horn, B. and Schunck, B.** (1981). *Determining Optical Flow*. Artificial Intelligence. pp. 185-203.
- JBenchmark.,** (2009). *JBenchmark*. [Online]. Available at: <http://www.jbenchmark.com>, (Accessed August 2009).

APPENDICES

- Jenabi, M. and Reiterer, H.** (2008). *Finteraction - Finger Interaction with Mobile phones*. NordiCHI 2008.
- Jones, Matt and Marsden, Gary.** (2006). *Mobile Interaction Design*, John Wiley & Sons Ltd, Chichester England.
- Kato, H and Billinghamurst, M.,** (1999). *Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System* . Proceedings of the 2nd International Workshop on Augmented Reality San Francisco.
- Kaxiras, S. Narlikar, G. Berenbaum, A. and Hu, Z.** (2001). *Comparing power consumption of an SMT and a CMP DSP for mobile phone workloads* . : International Conference on Compilers, Architecture and Synthesis for Embedded Systems.
- Kerr, S., Foster, G. and Thinyane, H.** (2008). *Vision Based Interaction Techniques for Mobile Phones: Current Status and Future Directions* . Southern Africa Telecommunication Networks and Applications Conference '08. Durban.
- Kerr, S., Thinyane, H. and Foster, G.** (2009). *Mobile Phone Performance Analysis for Camera Based Visual Interactions* . Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, Sasolburg.
- Kjeldskov, J. and Stage, J.** (2004). *New Techniques for usability evaluation of mobile systems* . Elsevier Ltd. Volume 60, Issues 5-6, pp. 599 - 620.
- Knudsen, J.** (2007) *Taking Pictures with MMAPI / Sun Microsystems Development*. [Online]. Available at: <http://developers.sun.com/mobility/midp/articles/picture/index.html> (Accessed April 2008).
- Koller, D., Weber, J., Huang, T., Malik, J., Ogasawara, G., Rao, B. and Russell, S.** (1994). *Towards Robust Automatic Traffic Scene Analysis in Real-time* . Pattern Recognition, Vol. 1. Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on Artificial Intelligence.
- Koprowski, G.** (2007). *Research: Mobile Phone Industry Poised for Spectacular Growth*. [Online]. Available at: <http://www.technewsworld.com/story/48423.html> (Accessed March 2008).
- Lai, W. and Duh, B.** (2004). *Effects of frame rate for visualization of dynamic quantitative information in a head-mounted display* . IEEE International Conference: Systems, Man and Cybernetics. Volume 7.

APPENDICES

- Lattimore, P.** (2007). *Gadgets*. [Online]. Available at: <http://www.tech.co.uk/gadgets/phones/mobile-phones/news/nokia-touchscreen-iphone-rival-due-next-year?articleid=1963812170> (Accessed May 2008)
- Lenman, S., Bretzner, L. and Thuresson, B.** (2002). *Computer Vision Based Hand Gesture Interfaces for Human-Computer Interactions* . CID '02. TRITA-NA-D0209, CID-172.
- Letessier, J. and Bérard, F.** (2004). *Visual Tracking of Bare Fingers for Interactive Surfaces* .UIST '04. Santa Fe, USA.
- Love, S.** (2005). *Understanding Mobile Human-Computer Interaction*. Elsevier Butterworth-Heinemann, Architectural Press, Oxford.
- Lin, M., Goldman, R., Price K., Sears, A. and Jacko, J.** (2007). *How do people tap when walking? An empirical investigation of nomadic data entry* . International Journal of Human-Computer Studies. Volume 65, Issue 9. pp 759 - 769.
- Liu, H., Hong, T., Herman, M., Camus,T. and Chellappa, R.** (1998). *Accuracy vs Efficiency: Tradeoffs in Optical Flow Algorithms*. Computer Vision and Image understanding. Vol. 72, Issue 3. pp. 271 - 286.
- Long, A. Landsay, J. and Rowe, L.** (1999). *Implications For a Gesture Design Tool* . SIGCHI '99. Berkeley.
- Ludwig, C. and Reimann, C.** (2005). *Augmented Reality: Information at Focus* . C-Lab Report Vol.4, No. 1.
- Lund, A.** (1998). *The Need for a Standardized Set of Usability Metrics*, Human Factors and Ergonomics Society Conference. Chicago.
- Martin, J.** (1998). *TYCOON: Theoretical Framework and Software Tools for Multimodal Interfaces* . In **John Lee** (Ed.) Intelligence and Multimodality in Multimedia Interfaces.
- Martin, J., Devin, V. and Crowley, J.** (1998). *Active Hand Tracking* . GRAVIR - IMAG.
- McCane, B., Novins, K., Crannitch, D. and Galvin, B.** (2001). *On Benchmarking Optical Flow*. British Machine Vision. pp 126-143.
- McNeil, D., Levy E. and Pedelty, L.** (1992). *Hand and Mind: What Gestures Reveal About Thought*. University of Chicago Press. Chicago

APPENDICES

- Microsoft.,** (2005). *Windows XP/ What is a Tablet PC.* [Online]. Available at: <http://www.microsoft.com/windowsxp/tabletpc/evaluation/about.msp> (Accessed September 2008).
- Mitran, M.,** *Literature Review on Optical Flow.* McGill University, Montreal.
- Moeslund, T. and Granum, E.** (2001). *A Survey of Computer Vision-Based Human Interactions.* Computer Vision and Image Understanding Vol. 81, pp. 231–268. 1077-3142/01.
- Mulder, A.** (1996). *Hand Centered Studies of Human Movement Project: Hand Gestures for HCI,* Simon Fraser University.
- Nokia.** (2008). *Get Started with Mobile Development .* [Online]. Available at: http://www.forum.nokia.com/I_Want_To/Develop_Mobile_Applications/Get_Started.xhtml#professional_developer (Accessed July 2009).
- Nokia.** (2003). *Introduction to Mobile Game Development .* [Online]. Available at: http://www.forum.nokia.com/info/sw.nokia.com/id/3f4d7e31-f200-46db-8f10-611f953485e8/Mobile_Game_Intro_v1_1.pdf.html (Accessed September 2008).
- Nokia.** (2008). *Nokia leads GSM Evolution in Americas with new GPRS handsets .* [Online]. Available at: <http://www.nokia.com/A4136002newsid=852457> (Accessed January 2009).
- Nokia.** (2009). *Python for S60.* [Online]. Available at: <http://wiki.opensource.nokia.com/projects/PyS60> (Accessed March 2009)
- NokiaResearchCenter.** (2008). *class_c_camus.* [Online]. Available at: <http://research.nokia.com> (Accessed March 2008).
- Norman, D.** (1988). *The Psychology of Everyday Things.* Basic Books
- Oviatt, S.** (2002). *Multimodal Interfaces .* Lawrence Erlbaum, New Jersey.
- Paelke, V., Reimann, C. and Stichling, D.** (2004). *Kick-Up-Menus .* ACM CHI '04 extended abstracts on Human factors in computing systems. New York, USA.
- Pham, M. and Schwock, J.** (2007). *Quantitative image analysis of immunohistochemical stains using CMYK color model .* Doagn Pathol, 2:8.
- Pirhonen, A., Brewster, S. and Holguin, C.** (2002). *Gestural and Audio Metaphors as a Means of Control for Mobile Devices.* CHI 02, Minneapolis.

APPENDICES

- Reimann, C and Paelke, V.** (2006). *Computer Vision based Interaction Techniques for Augmented Reality* . Universität Paderborn, Universität Hannover, Institut für Kartographie und Geoinformatik.
- Research On Asia Group.** (2006). *Camera Phone-based Mobile Search* : White Paper. ROA Group, Tokyo.
- Rahmati, A. and Zhong, L.** (2008). *Human-Battery Interaction on Mobile Phones*. Pervasive and Mobile Computing 5. pp. 465 - 477. Elsevier B. V. Ltd.
- Rohs, M.** (2007). *Marker-Based Embodied Interaction for Handheld Augmented Reality Games*. Journal of Virtual Reality and Broadcasting, Volume 4, No. 5.
- Rohs M., Gfeller, B. and Friedemann, M.** (2006). *Visual Code Recognition for Camera-Equipped Mobile Phones* . ETH 06, Zurich.
- Schmalstieg, D and Wagner, D.** (2007). *Experiences with Handheld Augmented Reality*. ISMAR '07 keynote paper.
- Shapiro L. and Stockman G.** (2002). *Computer Vision*. Prentice Hall. ISBN 0-13-030796-3.
- Sharma, R., Pavlovic, V. and Huang, T.** (1998). *Toward multimodal human-computer interface* . Proceedings IEEE, 86 [5] (Special issue on Multimedia Signal Processing).
- Sharp, R.** (2004). *Overview: New Uses for Camera Phones*. Technology@Intel Magazine. July 1.
- Shen, J and Castan, S.** (1992). *An Optimal Linear Operator for step edge detection*. Computer Vision, Graphics and Image Processing '92 (54).
- Sielhorst, T., Sa, W., Khamene, A., Sauer, F., and Navab, N.** (2007). *Measurement of absolute latency for video see through augmented reality*. International IEEE and ACM Symposium on Mixed and Augmented Reality '07. 10.1109/ISMAR.2007.4538850.
- Song, P., Yu H. and Winkler S.** (2009). *Vision-based 3D Finger Interactions for Mixed Reality Games with Physics Simulation*. The International Journal of Virtual Reality '09. - 8(2), pp. 1 - 6.
- Srihari, S.** (1988). *Analysis of Textual Images using the Hough Transformation*. Technical Report, State University of New York, Buffalo.
- Streeter, L.** (2001). *A Direct Geometric Algebra Optical Flow Solution*. University of Waikato, New Zealand.

APPENDICES

- Sturman, D.** (1992), *Whole Hand Input*. Ph.D. Thesis. Massachusetts Institute of Technology Cambridge, MA.
- Symbian.** (2008). *Symbian Publishes Q1 2008 Results*. [Online]. Available at: http://www.tracyandmatt.co.uk/blogs/index.php/symbian_publishes_q1_2008_resultsa_8207 (Accessed December 2009)
- Tullis, T. and Stetson, J.** (2004). *A Comparison of Questionnaires for Assessing Website Usability*. Usability Professionals Association, 2004, pp. 7 - 11.
- Wagner, D and Schmalstieg, D.** (2003). *ARToolkit on the PocketPC Platform*. Proceedings of the 2nd IEEE International Augmented Reality Toolkit Workshop '03, pp 14-15, Japan, Tokyo.
- Wagner, D and Schmalstieg D.** (2007). *Artoolkitplus for pose tracking on mobile phones*. Proceedings of the 12th Computer Vision Workshop '07.
- Wang, J. and Canny, J.** (2006). *TinyMotion: camera phone based interaction techniques*. Conference on Human Factors in Computing Systems '06.
- Wang, J., Zhai, S. and Canny, J.** (2006). *Camera Phone Based Motion Sensing: Interaction Techniques, Applications and Performance Study*. UIST '06, Montreux. ACM 1-59593-313-1/06/0010.
- Watson, R.** (1993). *A survey of Gesture Recognition Techniques*. Trinity College Press, Dublin.
- Wilhelm, A., Takhteyev, Y., Van House, N., Davis, M.** (2004). *Photo Annotation on a Camera Phone*. CHI '04, Vienna.
- Winkler, S., Rangaswamy, K. and Zhou, Z.** (2005). *Intuitive User Interface for Mobile Devices Based on Visual Motion Detection*. National University of Singapore, Singapore.
- Wirefly.** (2008). *Wirefly Survey Shows Cell Phones Are Growing As Camera Of Choice*. [Online]. Available at: http://www.wirefly.com/learn/company_news/cell-phones-are-growing-as-camera-of-choice-wirefly-survey-shows/ (Accessed March 2009).
- Wisniewski, D., Morton, D., Robbins, B., Welch, J., DeBenedictis, S., Dunin, E., Estanistao, J., James, D., Mills, G., Walton, G. and Valadares, J.** (2005). *Mobile Games White Paper*. IGDA '05.
- Wolfe, A.** (2003). *Microsoft's Compact Framework Targets Smart Devices*, ACM Queue, Vol. 1.

APPENDICES

- Yakhnin, A.** (2008). *How to Create a Microsoft .NET Compact Framework-based Image Button*, Windows Embedded Developer Center. [Online]. Available at: <http://msdn.microsoft.com/en-us/library/aa446518.aspx> (Accessed September 2008).
- Yamashita, A., Barendregt, W. and Morten, F.** (2007). *Exploring Potential Usability Gaps when Switching .* - British Computer Society : People and Computers XXI – HCI : Proceedings of HCI '07.
- Zerweck, P.** (1999). *Multidimensional Orientation Systems in Virtual Space on the basis of Finder .* Lawrence Erlbaum Associates, Munich.

Appendix A

Please circle the value you think is appropriate for each of the questions ranked from 1(Very Low) to 10 (Very High)

Steering Game

	Keypad Interaction	Camera Finger Interaction
	Very Low Very High	Very Low Very High
How mentally demanding was the task?	1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10
How physically demanding was the task? i.e difficulty in positioning and maintaining fingers in front of camera	Very Low Very High 1 2 3 4 5 6 7 8 9 10	Very Low Very High 1 2 3 4 5 6 7 8 9 10
How hurried or rushed was the pace of the task?	Well Hurried Paced 1 2 3 4 5 6 7 8 9 10	Well Hurried Paced 1 2 3 4 5 6 7 8 9 10
How successful were you in accomplishing what you were asked to do? Do you feel you got a good score?	Very Low Very High 1 2 3 4 5 6 7 8 9 10	Very Low Very High 1 2 3 4 5 6 7 8 9 10
How hard did you have to work to accomplish your level of performance?	Very Low Very High 1 2 3 4 5 6 7 8 9 10	Very Low Very High 1 2 3 4 5 6 7 8 9 10
How discouraged, irritated, stressed, and annoyed were you?	Very Low Very High 1 2 3 4 5 6 7 8 9 10	Very Low Very High 1 2 3 4 5 6 7 8 9 10

Overall for this task which method would you prefer to use? (circle one)

- Keypad
- Camera

Would you like to add anything else?

APPENDICES

Please circle the value you think is appropriate for each of the questions ranked from 1(Very Low) to 10 (Very High)

Target Acquisition

	Keypad Interaction					Camera Finger Interaction														
	Very Low					Very High														
How mentally demanding was the task?	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
How physically demanding was the task? i.e difficulty in positioning and maintaining fingers in front of camera	Very Low					Very High					Very Low					Very High				
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
How hurried or rushed was the pace of the task?	Well Hurried					Paced					Well Hurried					Paced				
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
How successful were you in accomplishing what you were asked to do?	Very Low					Very High					Very Low					Very High				
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
How hard did you have to work to accomplish your level of performance?	Very Low					Very High					Very Low					Very High				
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
How discouraged, irritated, stressed, and annoyed were you?	Very Low					Very High					Very Low					Very High				
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10

Overall for this task which method would you prefer to use? (circle one)

- Keypad
- Camera

Would you like to add anything else?

APPENDICES

Please circle the value you think is appropriate for each of the questions ranked from 1(Very Low) to 10 (Very High)

Directional Gestures

	Keypad Interaction					Camera Finger Interaction														
	Very Low					Very High														
How mentally demanding was the task?	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
How physically demanding was the task? i.e difficulty in positioning and maintaining fingers in front of camera	Very Low					Very High					Very Low					Very High				
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
How hurried or rushed was the pace of the task?	1	2	3	4	5	6	7	8	9	10	Well Paced					Well Hurried				
	Well Paced					Well Hurried					Well Paced					Well Hurried				
How successful were you in accomplishing what you were asked to do?	1	2	3	4	5	6	7	8	9	10	Very Low					Very High				
	Very Low					Very High					Very Low					Very High				
How hard did you have to work to accomplish your level of performance?	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
	Very Low					Very High					Very Low					Very High				
How discouraged, irritated, stressed, and annoyed were you?	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
	Very Low					Very High					Very Low					Very High				

Overall for this task which method would you prefer to use? (circle one)

- Keypad
- Camera

Would you like to add anything else?

APPENDICES

Please place an X in the square which you most agree with ranked from 1 (Strongly Disagree) to 7 (Strongly Agree)

Usefulness	Strongly Disagree				Strongly Agree		
	1	2	3	4	5	6	7
• It helps me be more effective	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• I think that I would like to use this interaction technique frequently	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• It is useful	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• It makes the things I want to accomplish easier to get done	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• It saves me time when I use it	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• It meets my needs	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• It does everything I would expect it to do	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ease of Use	Strongly Disagree				Strongly Agree		
• It is easy to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• It is simple to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• It is user friendly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• It is flexible	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Using it is effortless	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• I felt very confident using the interaction technique	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• I can use it without written instructions	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• I don't notice any inconsistencies as I use it	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Both occasional and regular users would like it	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• I can recover from mistakes quickly and easily	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• I think that I would need the support of a technical person to be able to use this interaction technique	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ease of Learning	Strongly Disagree				Strongly Agree		
• I learned to use it quickly and easily	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• I easily remember how to use it	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• I quickly became skillful with it	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• I would imagine that most people would learn to use this technique quickly	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Satisfaction	Strongly Disagree				Strongly Agree		
• It is pleasant to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• I would recommend it to a friend	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• It is fun to use	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• It works the way I want it to work	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• It is wonderful	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• I feel I need to have it	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

APPENDICES

Camera Based Finger Interactions User Study

This user study hopes to analyze your experience with a new mobile interaction technique. We are testing this against the traditional keypad interaction technique found on most mobile phones. There are three tasks which we will ask you to perform. Each task will be completed using the keypad technique, and the finger interaction technique. After you have finished each task with each of the interaction techniques you will be asked to fill in the appropriate part of the questionnaire. Once all three tasks have been completed you will need to fill in the final section located on the last page of the questionnaire.

Each of the tasks will present you with a screen showing a camera viewfinder similar to what you would see when you are trying to take a picture. There may be some shapes overlaid on this viewfinder. Ignore them for now. Place your finger in the centre of the screen and press the 'OK' button. A blue box will flash up on the screen. This box is a calibration guide. Note that there is a bar at the top of the screen which shows the current colour which must be tracked. This will change based on whatever colour is placed in front of the camera and the 'OK' button is pressed. You can press the 'OK' button as many times as you wish. This will simply recalibrate the colour and restart the timer for the task. Once you have calibrated your finger colour you will be presented with one of the following tasks in no particular order:

Directional Gestures

You will be presented with a screen containing a series of coloured block. In order to move to the next block you must move your finger to the right to move the blocks to the right, or move your finger to the left of the screen to move the blocks to the left. You will notice that the writing at the top of the screen changes to the name of the coloured block which is currently centered and this is what we mean when we say a block is selected.

Your Task:

Select the **green** block >> Now select the **blue** block by moving **right** >> **Green** block by moving **right** >> **Blue** block moving **left** >> **Green** block by moving **left** >> **Red** block moving **left**>> and finally the **green** block moving **right**

APPENDICES

- Green
- Blue | right
- Green | right
- Blue | left
- Green | left
- Red | left
- Green | right

Target Acquisition

Four Targets are placed on the screen at equal distance from each other. The must move your finger to acquire each target on the screen by simply hitting the target with the pointer drawn on your finger or the purple pointer you are moving around with the keypad.

Your Task:

Hit all the targets in a **clockwise** order starting with the top right block. Do this **five** times.

Steering Game

The game requires you to move your finger to catch objects falling from the top of the screen. When an object is caught it changes colour, is reset and the score increased. Once the object has been caught it is out of play and is no longer worth any points. Once all the blocks have been caught you will receive a bonus and all blocks will change colour and be worth points again.

Your Task:

Try and get a **score** of **1000**.

APPENDICES

Appendix B: User Study Results

	Target Acquisition		Steering game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score	15	22	26	21	18	17	Ease of Use
Time	37.3	35.63	59.90	46.37	29.89	30.85	Ease of Learning
Misses	0	3	19	14	1	1	Satisfaction
Preferred	1	0	0	1	0	1	
Comments	Target Acquisition: I think that if I had more time to learn and get used to the camera technique that it would have been easier to use						
	Target Acquisition		Steering Game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score:	16	19	22	32	10	35	Ease of Use
Time	36.49	39.16	62.16	80.56	23.70	68.16	Ease of Learning
Misses	0	3	19	21	0	4	Satisfaction
Preferred	1	0	0	1	1	0	
Comments	Target Acquisition: although the camera was more entertaining Steering Game: The keypad took more effort than the camera Directional Gestures: The finger and speed factor made the task longer						
	Target Acquisition		Steering Game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score:	9	19	10	24	9	11	Ease of Use
Time	46.97	45.58	108.01	98.27	30.56	42.01	Ease of Learning
Misses	0	4	25	22	2	2	Satisfaction
Preferred	1	0	1	0	1	0	
Comments	None						
	Target Acquisition		Steering Game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score:	30	24	27	25	23	25	Ease of Use
Time	50.22	54.76	80.69	53.03	24.88	38.25	Ease of Learning
Misses	2	5	16	9	0	3	Satisfaction
Preferred	0	1	1	0	1	0	
Comments	Observer Comments: pressed menu for 4 secs, game						

APPENDICES

	Target Acquisition		Steering Game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score:	20	17	18	20	18	19	Ease of Use
Time	36.74	45.04	71.73	54.99	20.82	30.52	Ease of Learning
Misses	1	3	15	10	0	1	Satisfaction
Preferred	0	1	0	1	1	0	
Comments	Steering Game: The clicks and keypad take long compared to if a joystick was used						
	Target Acquisition		Steering Game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score:	30	32	31	28	27	30	Ease of Use
Time	39.72	40.37	46.67	42.37	29.07	31.61	Ease of Learning
Misses	0	4	14	10	0	1	Satisfaction
Preferred	1	0	0	1	1	0	
Comments	Target Acquisition: If it didn't lag when it disappeared at the bottom it would of been easier Directional Gestures: The pointer was slightly jumpy Observer Comments: problem with small movements. Wrong type of movements						
	Target Acquisition		Steering Game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score:	27	24	27	28	27	34	Ease of Use
Time	48.18	54.66	123.32	65.72	36.13	122.24	Ease of Learning
Misses	1	6	24	19	0	15	Satisfaction
Preferred	0	1	0	1	1	0	
Comments	Target Acquisition: Motion seems faster than the finger but I kept on having a problem at a corner General Comment: In time I might even prefer the finger interaction. It reminded me of early experiences with mouse and today I prefer mouse over arrow keys						
	Target Acquisition		Steering Game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score:	32	38	32	33	30	30	Ease of Use
Time			47.42	34.43	20.7	21.95	Ease of Learning
Misses			11	7	0	0	Satisfaction
Preferred	0	1	0	1	0	1	
Comments	Target Acquisition: Preferred camera because it is novel even though more						

APPENDICES

	frustrating Steering Game: Again due to novelty but also due to a better scroll speed with camera compared to keypad Directional Gestures: Novel entertaining to use						
	Target Acquisition		Steering Game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score:	28	41	35	45	28	46	Ease of Use
Time	33.09	47.49	49.71	63.61	32.74	35.16	Ease of Learning
Misses	0	3	12	19	0	0	Satisfaction
Preferred	1	0	1	0	1	0	
Comments	General Comments: Held finger too close to camera Observer Comments: nice idea having at back of device but no tactile feedback makes hard to use						
	Target Acquisition		Steering Game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score:	24	22	36	31	25	27	Ease of Use
Time	58.48	52.99	54.72	46.66	29.06	39.51	Ease of Learning
Misses	0	7	15	12	0	2	Satisfaction
Preferred	1	0	0	1	1	0	
Comments	Target Acquisition: I would prefer to use the camera if it followed your finger better and didn't freeze when your finger leaves the camera view Steering Game: Keypad to slow. Camera worked faster but took some time to get used to Directional Gestures: Took a while to calibrate camera setting was frustrating for the first few attempts. I did get the hang of it and the game became much easier Observer Comments: The user had long nails and this caused the system to lose finger sometimes, identifying the nail instead of the finger After the study the user tried applications again and was much faster Pressed menu button 10 seconds in game						
	Target Acquisition		Steering Game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score:	24	27	28	28	21	28	Ease of Use
Time	45.02	47.2	63.52	61.572	28.2	47.23	Ease of Learning
Misses	1	4	16	15	0	3	Satisfaction

APPENDICES

Preferred	1	0	0	1	1	0	
Comments	None						
	Target Acquisition		Steering Game		Directional Gestures		USE/SUS questionnaire
	Keypad	Camera	Keypad	Camera	Keypad	Camera	Usefulness
Q Score:	22	25	25	28	22	27	Ease of Use
Time	41	45.18	77.34	55.63	27.32	44.822	Ease of Learning
Misses	0	4	18	13	1	3	Satisfaction
Preferred	0	1	1	0	0	1	
Comments	None						
Overall Averages							
	Target Acquisition		Steering Game		Directional Gestures		
	Keypad	Camera	Keypad	Camera	Keypad	Camera	
Q Score:	23.08	25.8333	26.41666667	28.3333	21.5	27.4167	
Time	43.02	46.19	70.43	58.601	27.76	46.026	
Misses	0.454545455	4.18182	17	14.25	0.333333333	2.91667	
Preferred	7	5	4	8	9	3	
Comments							
Final Usefulness	29.7						
Final Ease of Use	46.7						
Final Ease of Learning	21.1						
Final Satisfaction	28						
Usefulness %	63.19149						
Ease of Use %	60.64935						
Ease of learning %	75.35714						
Satisfaction %	66.66667						

Appendix B

Accuracy Statistics

5	7	6		6		4	
6	6	6		6		5	
6	5	7		7		5	
4	5	6		7		5	
7	6	6		7		5	
4	6	7		6		6	
5	6	6		7		5	
6	5	7		7		6	
5	6	6		6		6	
6	7	7		7		5	
dark complex	normal complex	bright complex		Flourescent Complex		Tungsten Complex	
5.4	5.9	6.4		6.6		5.2	
ttest:	ttest:	ttest:					
0.212579892	0.014956364	0.138184754					
ttest dark comp and dark normal	ttest dark comp and bright comp	ttest dark normal and bright complex					
7	7	7		8		6	
5	8	8		7		7	
7	7	8		9		6	
8	8	8		7		7	
7	8	8		8		6	
7	7	7		8		6	
6	5	7		6		6	
6	8	8		8		7	
6	6	7		7		5	
7	7	8		8		6	
dark texture	normal texture	bright texture		Flourescent Texture		Tungsten Texture	
6.6	7.1	7.6		7.6		6.2	
ttest:	ttest:	ttest:					
0.212579892	0.00846815	0.052177242					
ttest dark texture normal texture	ttest dark texture bright texture						
6	8	7		8		7	
7	7	8		8		8	
8	8	8		9		7	

APPENDICES

7	8	8		8		7	
8	7	8		9		7	
7	8	8		8		8	
7	7	9		8		8	
7	8	8		7		7	
7	7	8		9		7	
8	8	8		10		7	
dark simple	normal simple	bright simple		Flourescent Simple		Tungsten Simple	
7.2	7.6	8		8.4		7.3	
ttest:	ttest:	ttest:					
0.167850656	0.003110428	0.167850656					
ttest dark simple normal simple	ttest dark simple bright simple	ttest normal simple bright simple					

Speed Benchmarks

Benchmarks	Python Ncv	C++	Java
FPS	10.05258753	18.8913182	0.189805699
FPS	11.18044747	17.44829876	0.550414968
FPS	9.385252377	18.19911	0.084936689
FPS	11.64786533	18.33809659	0.121040826
FPS	10.26300756	18.90995773	0.093248552
FPS	10.53497942	17.19654362	0.146549293
FPS	10.15296081	17.46211788	0.127625636
FPS	11.98059677	18.40111495	0.139750516
FPS	9.479248422	18.88290019	0.001073277
FPS	10.53654522	18.7	0.162652
delta time	0.091132872	0.05493	4.986585382
delta time	0.085906921	0.069	5.902402462
delta time	0.091010916	0.12	6.670882125
delta time	0.093442112	0.0432	6.050368346
delta time	0.089193606	0.0452	6.169802009
delta time	0.091769889	0.0315	6.2082509
delta time	0.098587594	0.043	5.445362285
delta time	0.099017067	0.05	5.273326658
delta time	0.089777607	0.0432	6.5303738
delta time	0.09249218	0.05747742	5.95668
Time to Track	10.2619527	0.0498	3.233396398
Time to Track	9.69614502	0.0543	4.477238012
Time to Track	9.505310073	0.06323	4.816617048
Time to Track	10.77159188	0.04123	3.155348027
Time to Track	9.24068529	0.03123	4.835082693
Time to Track	9.505756914	0.05234	3.306169131
Time to Track	9.237202744	0.05235	3.218960422

APPENDICES

Time to Track	9.589177865	0.04135	3.998565347
Time to Track	10.24776719	0.05132	4.18228555
Time to Track	9.920703125	0.0432258	3.895
Full Scan	21.30934422	0.02341	0.654630295
Full Scan	19.06080538	0.0134251	0.315763653
Full Scan	19.86783913	0.031324	0.237666949
Full Scan	20.39047883	0.00921345	0.469003909
Full Scan	21.33298824	0.01345	0.156326475
Full Scan	19.36678219	0.01234	0.101086158
Full Scan	21.59474554	0.00874257	0.090723571
Full Scan	20.94703674	0.01354	0.156529723
Full Scan	20.33113887	0.01011	0.453996884
Full Scan	20.2201563	0.012625	0.2702
Pixel Skip 10	0.100238724	0.01314	0.291072849
Pixel Skip 10	1.660477823	0.013423	0.191615324
Pixel Skip 10	1.945375343	0.0201324	0.294645251
Pixel Skip 10	1.300899806	0.01456	0.093140843
Pixel Skip 10	1.29932662	0.00934562	0.170149547
Pixel Skip 10	1.273443221	0.013425	0.181767681
Pixel Skip 10	0.952743961	0.011134	0.171123696
Pixel Skip 10	1.064921387	0.0093425	0.081069831
Pixel Skip 10	1.05568867	0.00921435	0.087885487
Pixel Skip 10	1.130625	0.01255787	0.1717
Pixel Skip 30	0.58372466	0.001415	0.141237759
Pixel Skip 30	0.326283781	0.0014	0.096706228
Pixel Skip 30	0.1323	0.0014	0.0432
Pixel Skip 30	0.205285189	0.000932145	0.278742828
Pixel Skip 30	0.283755917	0.001043	0.256098646
Pixel Skip 30	0.151474851	0.0008452	0.130931323
Pixel Skip 30	0.277554055	0.001001	0.010348192
Pixel Skip 30	0.853484302	0.0008934	0.166103895
Pixel Skip 30	0.269003446	0.001434	0.16027467
Pixel Skip 30	0.3453125	0.001058097	0.1469
Pixel Skip 30	0.19055847	0.000393425	0.139082112
Pixel Skip 50	0.297446622	0.0004342	0.086291247
Pixel Skip 50	0.160832118	0.000512345	0.133625695
Pixel Skip 50	0.177414401	0.000334156	0.151495018
Pixel Skip 50	0.213179117	0.000445	0.214606554
Pixel Skip 50	0.145542061	0.000532	0.116011673
Pixel Skip 50	0.294547447	0.0003424	0.128546107
Pixel Skip 50	0.191270207	0.00043215	0.160757319
Pixel Skip 50	0.247822911	0.000434213	0.217144215
Pixel Skip 50	0.215625	0.00044258	0.1428

APPENDICES

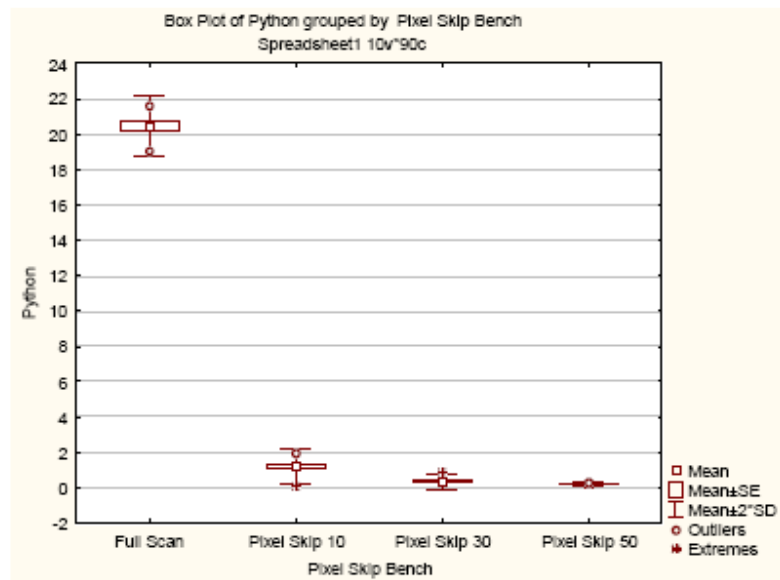
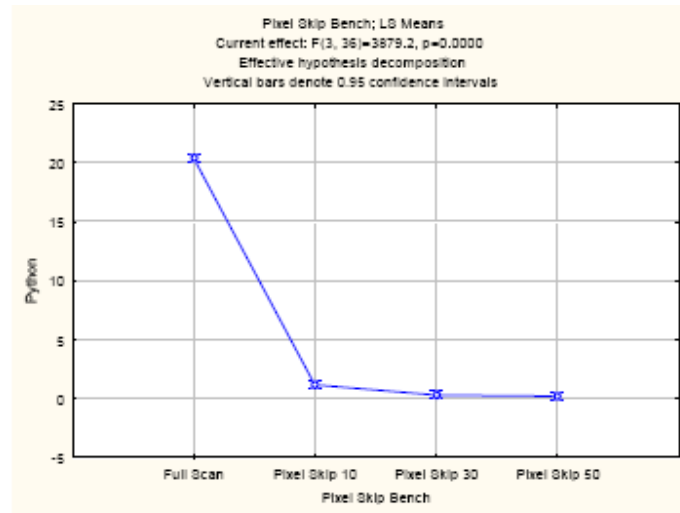
Latency	1.933681596	0.052	0.560759395
Latency	2.55675946	0.05432	0.966630684
Latency	2.174935915	0.051	0.34202258
Latency	2.306225786	0.045346	0.434537071
Latency	1.468867019	0.06345	0.399774459
Latency	2.393715688	0.05324	0.385411241
Latency	2.32760584	0.04245	0.312879259
Latency	2.583187635	0.062325	0.257639196
Latency	2.404638451	0.0501	0.453351677
Latency	2.3560182	0.05275	0.5100323
Dropped Frames	23.61980802	0.897934982	0.09012
Dropped Frames	22.8430508	0.890605096	0.08234
Dropped Frames	23.22668961	0.93344659	0.073425
Dropped Frames	23.24095909	0.964093793	0.083456
Dropped Frames	23.51264487	0.983018347	0.0734
Dropped Frames	22.42360335	0.865847756	0.08345
Dropped Frames	23.72746906	0.953766752	0.09345
Dropped Frames	23.25590829	0.851875177	0.094235
Dropped Frames	23.85767523	0.978127486	0.085
Dropped Frames	22.66175551	0.94645372	0.07379341

ANOVA Results

Effect	Univariate Tests of Significance for Python (Spreadsheet1) Sigma-restricted parameterization Effective hypothesis decomposition				
	SS	Degr. of Freedom	MS	F	p
Intercept	1222.050	1	1222.050	4797.035	0.00
Pixel Skip Bench	2964.660	3	988.220	3879.159	0.00
Error	9.171	36	0.255		

Cell No.	Pixel Skip Bench	Tukey HSD test; variable Python (Spreadsheet1) Approximate Probabilities for Post Hoc Tests Error: Between MS = .25475, df = 36.000			
		{1}	{2}	{3}	{4}
1	Full Scan	20.442	1.1784	.32898	.21596
2	Pixel Skip 10	0.000159	0.000159	0.002613	0.001191
3	Pixel Skip 30	0.000159	0.002613	0.959041	
4	Pixel Skip 50	0.000159	0.001191	0.959041	

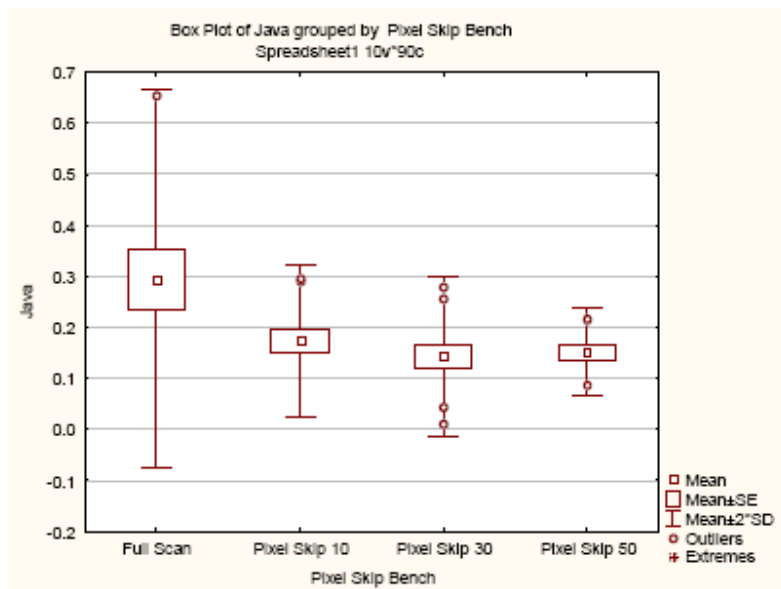
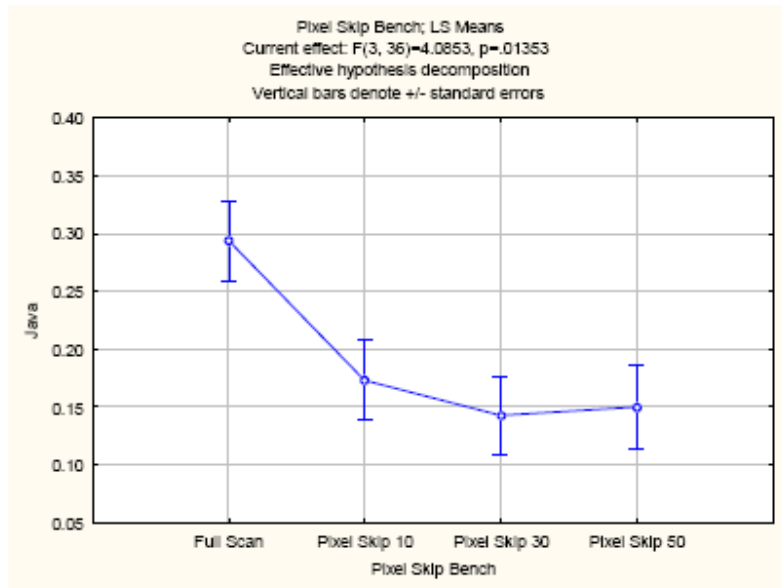
APPENDICES



Univariate Tests of Significance for Java (Spreadsheet1)					
Sigma-restricted parameterization					
Effective hypothesis decomposition					
Effect	SS	Degr. of Freedom	MS	F	p
Intercept	1.436158	1	1.436158	118.1490	0.000000
Pixel Skip Bench	0.148978	3	0.049659	4.0853	0.013530
Error	0.437597	36	0.012155		

APPENDICES

Tukey HSD test; variable Java (Spreadsheet1)					
Approximate Probabilities for Post Hoc Tests					
Error: Between MS = .01216, df = 36.000					
Cell No.	Pixel Skip Bench	{1}	{2}	{3}	{4}
		.29359	.17342	.14269	.15014
1	Full Scan		0.088086	0.017373	0.036283
2	Pixel Skip 10	0.088086		0.919144	0.967389
3	Pixel Skip 30	0.017373	0.919144		0.998839
4	Pixel Skip 50	0.036283	0.967389	0.998839	

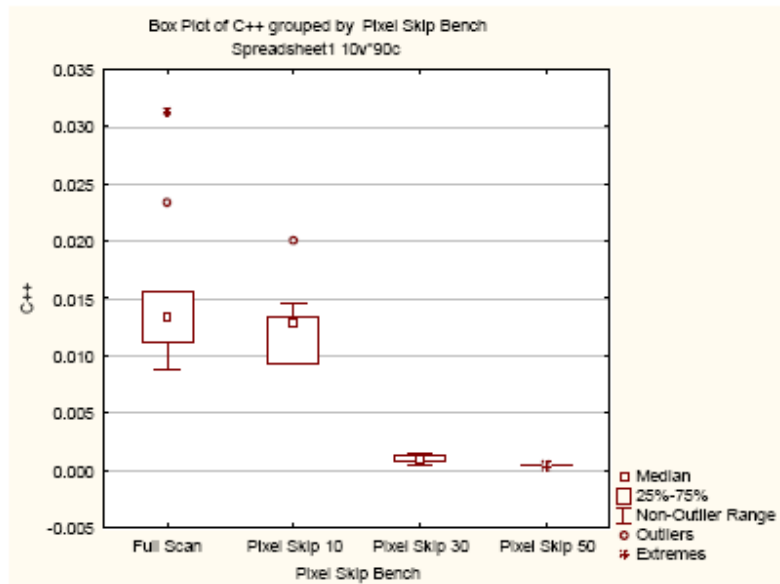
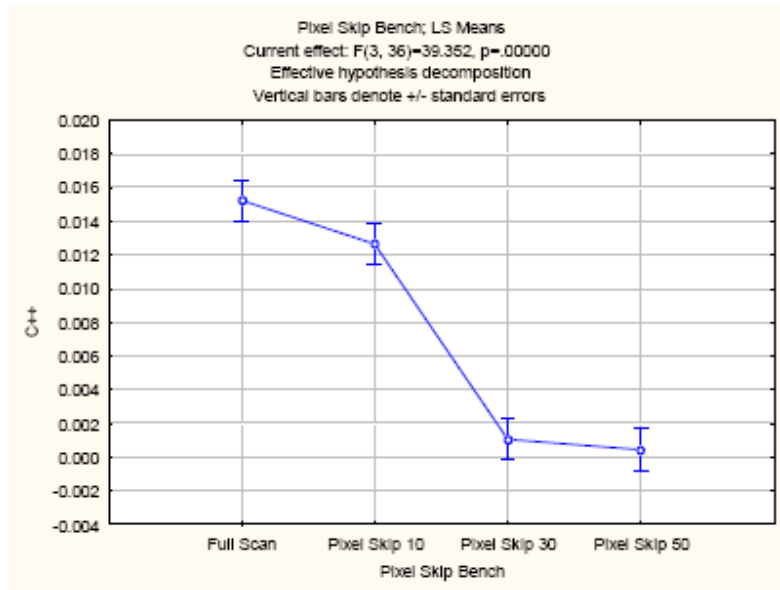


APPENDICES

Effect	Univariate Tests of Significance for C++ (Spreadsheet1) Sigma-restricted parameterization Effective hypothesis decomposition				
	SS	Degr. of Freedom	MS	F	p
Intercept	0.002143	1	0.002143	143.6636	0.000000
Pixel Skip Bench	0.001761	3	0.000587	39.3518	0.000000
Error	0.000537	36	0.000015		

Cell No.	Tukey HSD test; variable C++ (Spreadsheet1) Approximate Probabilities for Post Hoc Tests Error: Between MS = .00001, df = 36.000				
	Pixel Skip Bench	{1}	{2}	{3}	{4}
		.01522	.01263	.00107	.00043
1	Full Scan		0.448295	0.000159	0.000159
2	Pixel Skip 10	0.448295		0.000159	0.000159
3	Pixel Skip 30	0.000159	0.000159		0.982695
4	Pixel Skip 50	0.000159	0.000159	0.982695	

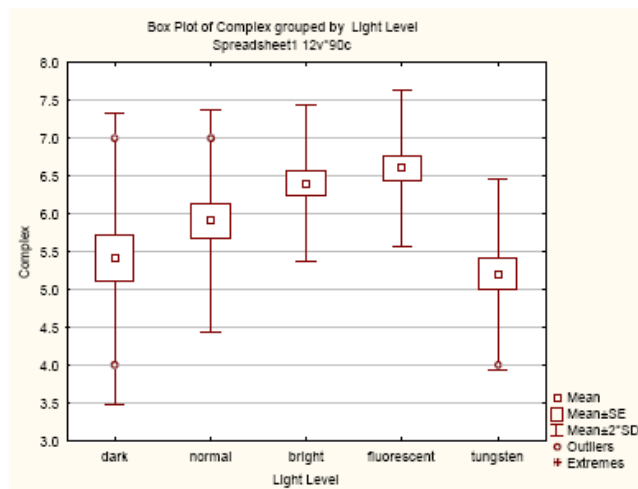
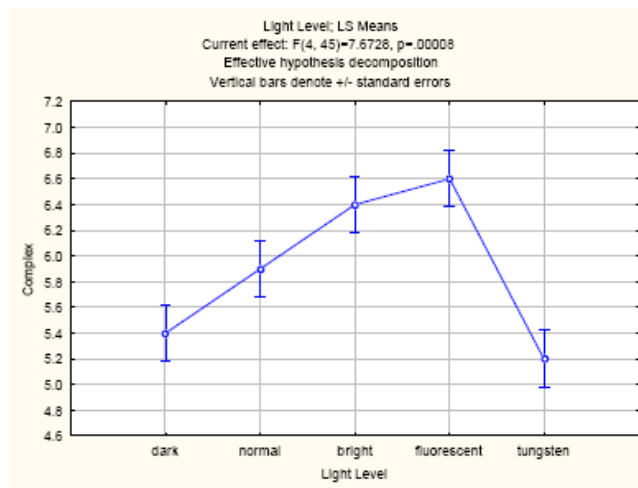
APPENDICES



Univariate Tests of Significance for Complex (Spreadsheet1) Sigma-restricted parameterization Effective hypothesis decomposition					
Effect	SS	Degr. of Freedom	MS	F	p
Intercept	1740.500	1	1740.500	3609.332	0.000000
Light Level	14.800	4	3.700	7.673	0.000084
Error	21.700	45	0.482		

APPENDICES

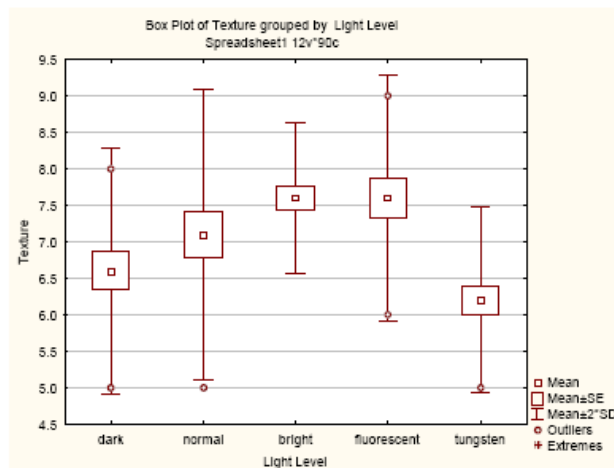
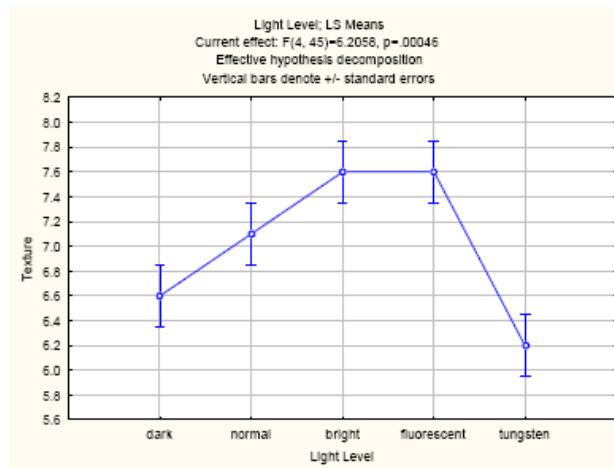
Tukey HSD test; variable Complex (Spreadsheet1) Approximate Probabilities for Post Hoc Tests Error: Between MS = .48222, df = 45.000						
Cell No.	Light Level	{1} 5.4000	{2} 5.9000	{3} 6.4000	{4} 6.6000	{5} 5.2000
1	dark		0.499040	0.019288	0.003254	0.966945
2	normal	0.499040		0.499040	0.179210	0.179210
3	bright	0.019288	0.499040		0.966945	0.003254
4	fluorescent	0.003254	0.179210	0.966945		0.000549
5	tungsten	0.966945	0.179210	0.003254	0.000549	



Univariate Tests of Significance for Texture (Spreadsheet1) Sigma-restricted parameterization Effective hypothesis decomposition					
Effect	SS	Degr. of Freedom	MS	F	p
Intercept	2464.020	1	2464.020	4002.921	0.000000
Light Level	15.280	4	3.820	6.206	0.000459
Error	27.700	45	0.616		

APPENDICES

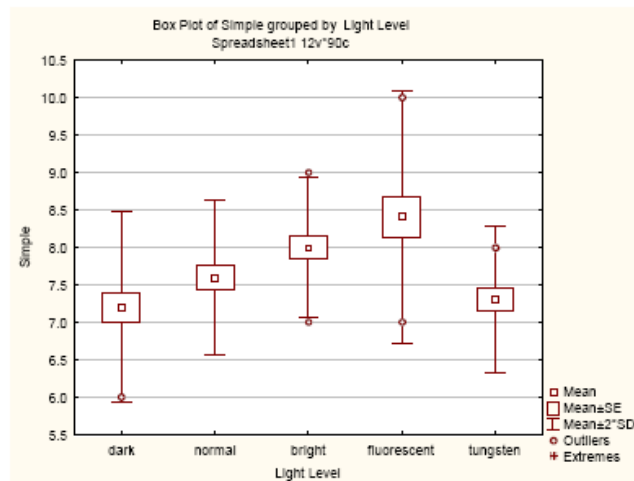
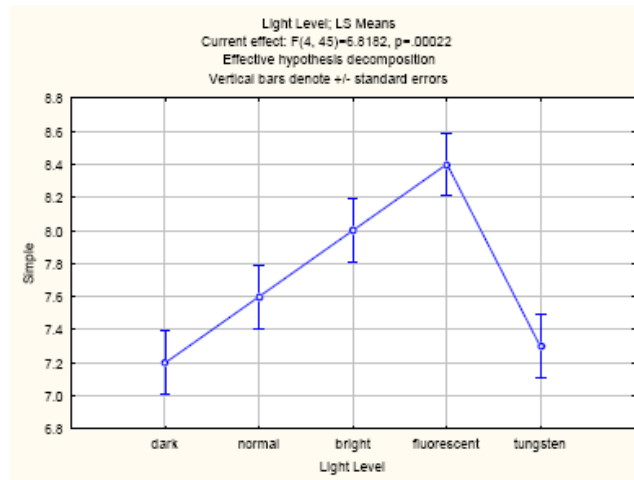
Tukey HSD test; variable Texture (Spreadsheet1)						
Approximate Probabilities for Post Hoc Tests						
Error: Between MS = .61556, df = 45.000						
Cell No.	Light Level	{1}	{2}	{3}	{4}	{5}
		6.6000	7.1000	7.6000	7.6000	6.2000
1	dark		0.615204	0.049078	0.049078	0.784581
2	normal	0.615204		0.615204	0.615204	0.094516
3	bright	0.049078	0.615204		1.000000	0.002248
4	fluorescent	0.049078	0.615204	1.000000		0.002248
5	tungsten	0.784581	0.094516	0.002248	0.002248	



Univariate Tests of Significance for Simple (Spreadsheet1)					
Sigma-restricted parameterization					
Effective hypothesis decomposition					
Effect	SS	Degr. of Freedom	MS	F	p
Intercept	2964.500	1	2964.500	8085.000	0.000000
Light Level	10.000	4	2.500	6.818	0.000223
Error	16.500	45	0.367		

APPENDICES

Tukey HSD test; variable Simple (Spreadsheet1)						
Approximate Probabilities for Post Hoc Tests						
Error: Between MS = .36667, df = 45.000						
Cell No.	Light Level	{1}	{2}	{3}	{4}	{5}
		7.2000	7.6000	8.0000	8.4000	7.3000
1	dark		0.582374	0.038049	0.000665	0.995961
2	normal	0.582374		0.582374	0.038049	0.801606
3	bright	0.038049	0.582374		0.582374	0.090479
4	fluorescent	0.000665	0.038049	0.582374		0.001833
5	tungsten	0.995961	0.801606	0.090479	0.001833	



Appendix C

Mobile Phone Performance Analysis for Camera Based Visual Interactions

Simon Kerr
Computer Science
Rhodes University
South Africa
+27 46-6038291
g03k0704@campus.ru.a
c.za

Hannah Thinyane
Computer Science
Rhodes University
South Africa
+27 46-6038291
h.thinyane@ru.ac.za

Greg Foster
Information Systems
Rhodes University
South Africa
+27 46-603829
g.foster@ru.ac.za

ABSTRACT

Vision based technology such as motion detection has long been limited to the domain of powerful processor intensive systems such as desktop PC's and specialist hardware solutions. With the advent of much faster mobile phone processors and memory we are now seeing a plethora of feature rich software being deployed onto the mobile platform. Since these high powered smart phones are now equipped with cameras, it has become feasible to combine their powerful processors and the camera to support new ways of interacting with the phone. However, it is not clear whether or not these processor intensive visual interactions can in fact be run at an acceptable speed on current mobile handsets. In this paper we look at one of the most popular and widespread mobile smart phone systems; the Symbian S60 and benchmark the speed, accuracy and deployability of the three popular mobile languages. We test a pixel thresholding algorithm in, C++, Python and Java and rank them based on their speed within the context of intensive image based processing.

Categories and Subject Descriptors

D.3.0[Software]: Programming Languages *General*
B.8.0[Hardware]: Performance and Reliability *General*

General Terms

Performance, Measurement, Languages

Keywords

Performance Analysis, Mobile Phones, Camera Based Visual Interactions, Pixel Threshold Algorithm

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAICSIT'09, 12-14 October 2009, Riverside, Vanderbijlpark, South Africa.

Copyright 2009 ACM 978-1-60558-643-4...\$5.00.

Vision Based Interaction Techniques for Mobile Phones: Current Status and Future Directions

Simon B. Kerr, Greg Foster, and Hannah Slay

Abstract—In recent years mobile devices have been deployed with various new technologies, such as high quality cameras and the ability to support rich multimedia. Vision based technology such as motion detection has, until recently been limited to more powerful desktop devices. This paper lays out a brief review of these technologies with the aim of introducing the concept of vision based interaction on mobile devices and substantiate an implementation thereof. We conclude that mobile phones are currently being deployed with hardware and software which can support vision based interactions and which in the future could be widely deployed. We predict that in the near future vision based interactions such as gesture control will become prevalent and greatly enhance mobile devices.

Index Terms—computer vision, gesture tracking, human-computer interaction, mobile devices, mobile interaction.

S.B. Kerr is with the Computer Science Department, Rhodes University, Grahamstown, South Africa (phone: 046 6038291; fax: 046 6361915; e-mail: g03k0704@campus.ru.ac.za)

G. Foster is with the Information Systems Department, Rhodes University, Grahamstown, South Africa (e-mail: g.foster@ru.ac.za).

H. Slay is with the Computer Science Department, Rhodes University, Grahamstown, South Africa (e-mail: H.Slay@ru.ac.za)