An Architecture to Support Multi-Touch Collaborative Information Retrieval

Ivan Sams, Janet Wesson and Dieter Vogts
Department of Computing Sciences
Nelson Mandela Metropolitan University, P. O. Box 77000, Port Elizabeth 6031
Tel: +27 41 503 2247, Fax: +27 41 503 2831
email: Ivan.Sams@nmmu.ac.za, Janet.Wesson@nmmu.ac.za, Dieter.Vogts@nmmu.ac.za

Abstract-Collaborative Information Retrieval (CIR) is the process by which people working together can collaboratively search for, share and navigate through information. Computer support for CIR currently makes use of single-user systems. CIR systems could benefit from the use of multi-user interaction to enable more than one person to collaborate using the same data sources, at the same time and in the same place. Multi-touch interaction has provided the ability for multiple users to interact simultaneously with a multi-touch surface.

This paper presents a generalised architecture for multi-touch CIR applications. There are three main goals of the proposed architecture: to create hardware independence, to separate the gesture recognition and CIR information objects from the application code and to make use of an extensible gesture definition set to allow for application-specific operations. A prototype CIR tool based on this architecture is described which will be used to investigate the potential for multi-touch interaction techniques to effectively support CIR.

Index Terms—Computer Supported Cooperative Work, Collaborative Information Retrieval, Human-Computer Interaction, Multi-touch, Touch Screens.

I. Introduction

Collaborative Information Retrieval (CIR) involves searching for, retrieving and viewing data as a group of people, using documents as data sources [1]. This activity can be supported by computer systems and can thus form a branch of Computer Supported Cooperative Work (CSCW). Collaboration may occur simultaneously, with users working together, or users may work on individual machines, distributed using networks. Personal computers do not effectively support simultaneous collaboration as they are designed for single-user, single-input environments [2].

Multi-touch surface computing has recently seen a reduction in costs and increased frequency of use [3]. This, together with its inherently multi-user nature, suggests that multi-touch interaction could be a possible solution for effectively supporting simultaneous, co-located CIR.

Developers of CIR applications need to consider the application architecture to support multi-touch interaction. Currently, multi-touch application developers predominantly make use of Software Development Kits (SDKs) that are provided by the hardware manufacturers [4]. This has the major drawback that applications written using a SDK will only work on that specific hardware platform. Furthermore, the complexity of the gesture recognition algorithms needs to be separated from the application logic using a gesture recognition framework. A framework for gesture recognition can also be re-used for other CIR applications.

The aim of this paper is to investigate the requirements for a multi-touch CIR framework and to propose a suitable architecture that supports these requirements. Section II reviews related work in the fields of CIR and multi-touch interaction. The requirements of a framework to support CIR using multi-touch interaction techniques are investigated in Section III. Section IV presents the proposed architecture and Section V describes a prototype design that implements this architecture. The paper concludes with key findings in Section VI and future work in this area.

II. Related Work

This review of related work discusses literature in two main areas: Collaborative Information Retrieval, discussed in Section II-A, and Multi-touch Interaction, discussed in Section II-B.

A. Collaborative Information Retrieval

CIR may be defined as an information access activity that involves people interacting with other people in an information seeking and retrieval process. This may be directly between collaborators and with collaborators using documents as information sources [5]. Information sharing practices within a working environment can be broadly categorised into four types, namely strategic, paradigmatic, directive and social sharing. Directive sharing is practiced between colleagues in similar roles who share information in order to accomplish a task. Directive sharing is especially suited to computer support as it involves goal-oriented, two-way sharing [6].

CIR tasks can be divided into two categories: information seeking and information navigation. Information seeking can be defined as the process whereby information is found using querying and filtering tasks [7]. Studies have shown that collaboration is common in information seeking tasks. There are three main reasons for people to collaborate: working requirements, division of labour and diversity of skills [8]. Collaboration can improve the efficiency of achieving shared goals by delegating tasks and avoiding repetition, especially in the case of shared goals [1].

Paper documents are often the most intuitive way of sharing and annotating data. Working collaboratively at a traditional PC workstation does not provide intuitive collaborative interaction. This includes orientating the document to face another user, passing papers across a table, and making casual annotations [9]. However, electronic sources provide several benefits over physical documents. Electronic sources are easily copied, retrieved, distributed and backed up. They can be quickly searched and sorted with computer assistance. Different views are available to suit the user interface, such as minimizing to thumbnails or viewing documents extra-large, which is not possible with physical documents. Electronic sources often provide the most recent information, since the information space is dynamic and printed or published sources may be out of date.

Five existing CIR applications were analysed to determine the functional requirements of CIR applications: *Ariadne* [10], *SearchTogether* [11], *Coagmento* [12], *Cerciamo* [13], and *Annotate!* [14]. These requirements are analysed and presented in Section III.

B. *Multi-touch Interaction*

Touch screen input is a very direct form of Human-Computer Interaction (HCI). The input and output device is the same surface which makes it an appealing choice for interaction. The interface is simplified as no extra input devices are needed. Touch screen interfaces are also more robust than a free moving input device such as a mouse. Touch screen interfaces are frequently used in public computers as they are very intuitive to use, even when the user has no experience with the interface. They have found success in automated teller machines (ATMs) and information kiosks [15].

Single-touch screens have long been used as a mouse replacement. Multi-touch interaction has recently risen to prominence and is opening up new possibilities for novel, intuitive and natural interaction paradigms [4]. In particular it affords multi-user interaction which enables simultaneous collaboration as it can accept input from multiple users on the same device.

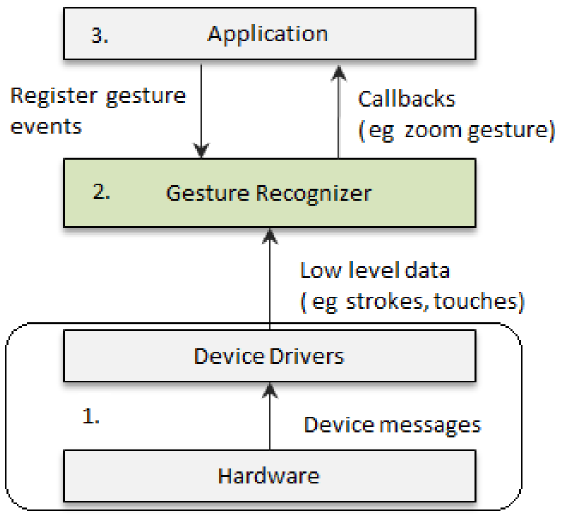


Figure 1: Typical gesture recognition process of a multi-touch system [4]

Multi-touch technology is not yet standardised and the events that are generated are specific per hardware platform. The Tangible User Interface Object (TUIO) protocol and the Windows Presentation Foundation (WPF) both go some way towards standardising these events but neither provides a user-interface level gesture processor or recognition engine [16]. Currently, multi-touch systems are not supported by a general, hardware-independent framework that separates the gesture recognition algorithms from the application logic. As a result, for each application developers are required to create their own architecture that is built on top of the device SDK [4]. A typical gesture processing architecture is shown in Figure 1. The application requires a gesture recognition engine, which has gesture events registered with it. The gesture recognition engine receives low level messages from the device drivers. From these messages the engine needs to generate high-level, per-object events that can be passed through to the application via gesture callbacks.

Several multi-touch frameworks have been recently proposed. These frameworks all provide support for basic gestures, and most provide for extending the gestures and visualisation of the gestures [17]. One such framework is the GestureToolkit (formerly TouchToolkit) [4]. The GestureToolkit enables developers to create multi-touch applications without having to create the gesture recognition engine or hard-code recognition algorithms. Four gesture definitions covering basic manipulation of objects are included in the toolkit: zoom, rotate, drag and lasso. The GestureToolkit also handles concurrency of interactions and the distribution of gesture events to the various User Interface (UI) elements. This framework makes use of a Gesture Definition Language (GDL). New gestures can be specified in the GDL which simplifies the creation of new gestures and hides the complex processing behind the gesture recognition engine. Input providers for several hardware platforms are supplied and may also be written by developers using the Toolkit.

III. Requirements Analysis

Table 1: Functional requirements of a general CIR application

#	Description	Freq.
1.	Provide shared access to an information space	5
2.	Search by collaboratively querying and filtering the information space	4
3.	Present or visualise the search results or shared information	5
4.	Collaboratively navigate through and retrieve search results	2
5.	Open documents to use as information sources	3
6.	Manipulate (e.g. order or categorise) the search results or shared information	2
7.	Update or add value to shared information	4
8.	Update users on other users' actions	3
9.	Enable communication between group members	2
10.	Enable division of workload between group members	3
11.	Store logs of user actions, such as communication and searches	3
12.	Enable users to refind and reuse information.	4

Computer support can facilitate CIR in several ways.

Information can be automatically or collaboratively queried, filtered and sorted based on the activities of a user or another collaborator [7]. In order to determine the common functional requirements of a CIR application, five existing applications were analysed to determine their functional requirements. A requirement was identified as being a general CIR functional requirement if it was present in 2 or more of the CIR applications. Table 1 gives the twelve requirements that were identified and the frequency of these requirements.

From a taxonomy and overview of multi-touch gesture frameworks, three requirements were determined for the CIR gesture processor [17]. These are:

1. **Hardware independence**: Applications built using this architecture should work on most multi-touch devices, and the architecture should allow for new devices to be added.
2. **Extensibility**: The framework should allow for developers to add application-specific gestures easily.
3. **Separation of application layers**: Gesture recognition and processing should occur separately from the application logic and user interface.

The CIR tasks need to be supported by a reusable library of multi-touch objects. These objects must support the twelve functional requirements identified in Table 1. In the next section an architecture will be proposed that incorporates the gesture processor and the multi-touch object library.

IV. Proposed Architecture

The SDK for the multi-touch hardware uses a set of native events which are generated when gestures are made on the screen. The events are atomic and a given gesture consists of a series of events. It is thus necessary to use a gesture processor to recognise gestures at the application level.

The SDK also generates WPF touch events. WPF touch events are created from the device information and provide a measure of hardware independence. WPF events cover most of the multi-touch functionality, although some hardware-specific functionality is not supported, such as pressure sensitivity and user identification. The architecture can function using WPF events as the hardware input, to take advantage of the level of hardware independence WPF provides. However, developers may choose to create a hardware-specific input provider. This is created using the SDK and native events of the hardware platform.

Figure 3 shows a schematic diagram of the proposed architecture. There are four main layers: the hardware layer, the gesture processor layer, the application layer and the CIR interaction layer. The first three layers correspond to the three layers of the gesture processing architecture described in Figure 1. The Gesture Toolkit was chosen as the gesture processor for the architecture. This toolkit satisfies the three requirements of the gesture processor: hardware independence, extensibility and separation of application layers that were identified in Section III. It is also extensible, allowing for a developer to create a specific input provider for this or another multi-touch device should the developer require hardware-specific functionality.

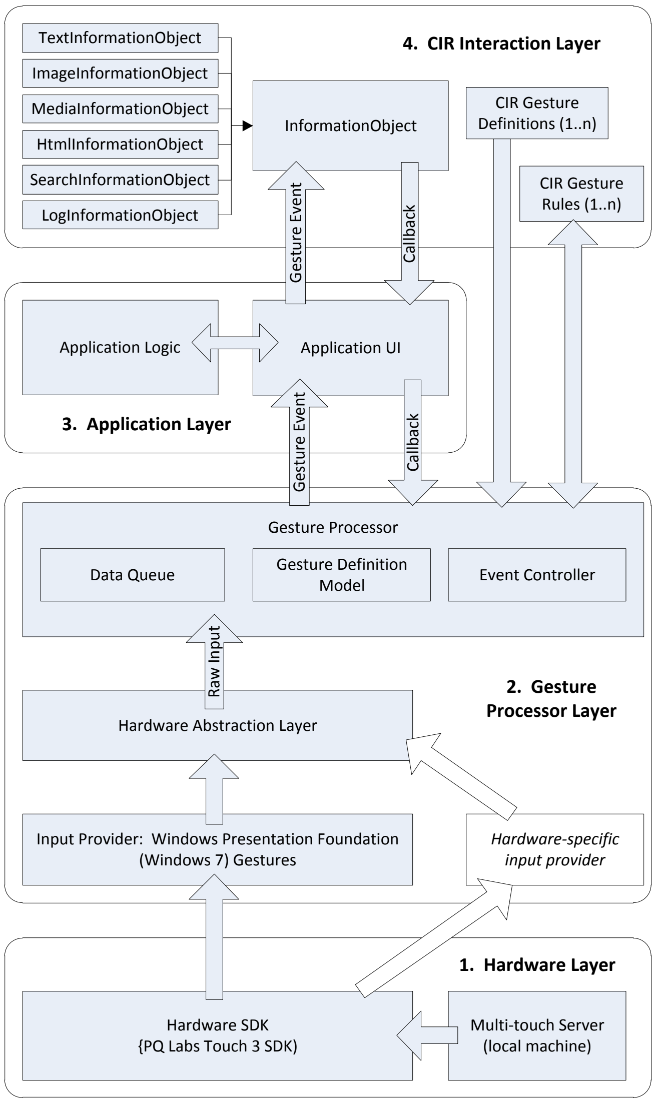


Figure 2: Proposed Architecture for CIR applications incorporating Gesture Toolkit

CIR gesture definitions and rules are specified as part of the CIR application framework. These are loaded by the Gesture Toolkit on demand, helping to utilise memory and processing power most efficiently. The gesture definition model is then built from the CIR gesture definitions and rules.

The gesture processor then evaluates the data queue against the primitive conditions given in each gesture definition. The framework stores each interaction per object to allow multiple users to interact simultaneously. It compares the data queue to the gesture definition model until it finds a match and recognises the gesture. The gesture processor calculates the requested return values, such as the location of the gesture and the size of the gesture. The event controller then generates the gesture event and passes it to the application UI.

When the application UI receives a gesture event, it bubbles the event up to the topmost UI object. The topmost object is the visible object and most likely the one the user intended to perform the gesture upon. This may be an information object, such as a document or image, or the workspace itself.

CIR applications need to provide access to an information space and provide a visualisation of search results. A library of information objects was created that extends the WPF controls to cater for functional requirements 3 to 7 of a CIR application (Table 1). The various information objects and the functionality they support are shown in Table 2.

Table 2: Information objects and functionality

InformationObject	Abstract base class providing general functionality: Select, Move, Zoom, Rotate, Duplicate, Annotate, Create link, Close
TextInformationObject	Presents text only information and files: Select text, Copy text
ImageInformationObject	Presents an image: Crop
HtmlInformationObject	Presents a document in HTML format: Select text, Copy text, Open hyperlink
MediaInformationObject	Presents an audio/visual object: Play/Pause, Seek, Volume, Copy snapshot
SearchInformationObject	Special object with which searches of the information space can be conducted. (Requirements 2 and 12 of Table 1): Search, Open result, Remove result
LogInformationObject	Special object which logs user actions (Requirements 8-11 of Table 1): Undo/Redo

The base class is `InformationObject` and provides attributes and methods common to all types of information object. This class is extended to provide specific information objects for four types of data and two special objects that help satisfy the other functional requirements. The four main types are Text, Image, Media and HTML objects (to display rich text, documents and web pages). These, or the InformationObject base class, can be easily extended to display any other types of information, such as Visio drawings and equations.

Table 3 shows a list of the functions available to the information object classes along with a description and diagram of the gesture that invokes the function. The Class column indicates which of the information object types the gesture applies to (Main being the workspace itself). Existing gesture conventions have been used for basic manipulation (Move, Zoom, Rotate and Select). For object-specific interaction, a two-handed approach was used. One hand holds the object whilst the other hand performs some function on the object, to avoid ambiguity with other gestures. To select text on an object for example, one would place two fingers from one hand on the object while the other selects the text.

V. Prototype Design

A prototype that makes use of the proposed architecture was implemented in order to evaluate the effectiveness of the architecture and interaction techniques. This prototype must satisfy all of the CIR functional requirements given in Section III.

The first requirement was to provide a shared access to an information space. This could be a database or a document collection for example. Three of the five CIR tools reviewed used the World Wide Web (WWW) as an information space. The WWW provides information on nearly every topic, and Web searches can be performed by interfacing with an existing search engine. The WWW was thus an obvious choice for the prototype information space. The prototype presents information objects created from Google search results, which can then be manipulated. As discussed in Section IV, the Information Object class library provides functionality to support the other functional requirements. These requirements include collaboratively navigating, querying and filtering the objects, as well as adding value to the objects in the form of annotations, interaction histories and links between objects.

The prototype UI presents a shared workspace to the collaborators. Unlike some other CIR applications such as *Cerchiamo* [13], each user assumes an equal role in the team, and any *ad hoc* special roles are user-mediated rather than system-mediated, such as group leader, Prospector (a user who searches a broad space) and Miner (a user who searches within the Prospector's results). Users may create search objects to search using an existing web search engine. The results may then be manipulated and distributed amongst the users automatically or manually. Value is added to the information objects in the form of automatic interaction logging, user-created annotations and user-created relationships between objects.

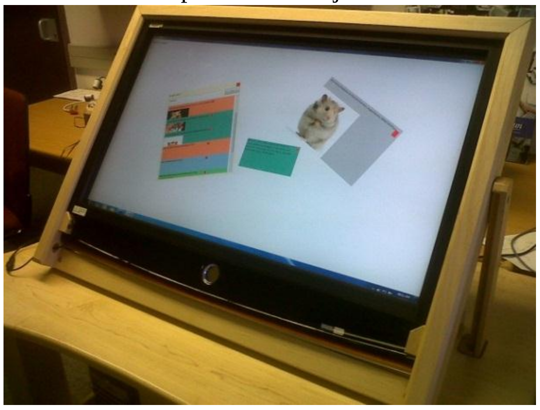
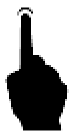
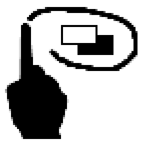
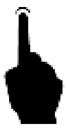
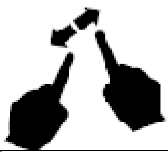
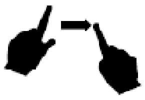
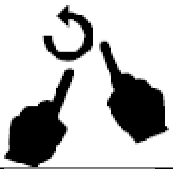
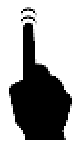
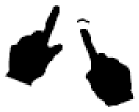
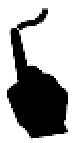
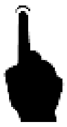
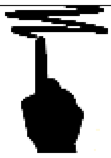
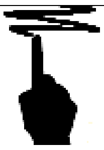
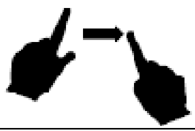
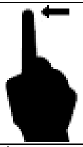
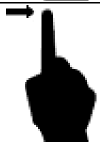


Figure 3: NMMU/Telkom CoE Multi-touch Surface and Driver PC with CIR prototype running

A 42" adjustable multi-touch surface belonging to the NMMU/Telkom CoE was used to implement the prototype (Figure 2). The device makes use of a PQ Labs G3 Multi-touch overlay to provide the touch input, which can recognise up to 32 touch points simultaneously [19].

The implementation of the prototype is currently being finalised. To evaluate the prototype, teams of users will be required to perform several CIR tasks. These will include performing a search, retrieving search results and distributing them amongst the team, manipulating the objects, creating annotations and relationships between objects and ultimately retrieving information. The effectiveness of the prototype will be evaluated by measuring the time taken to complete the tasks and asking the users to report on their user satisfaction.

Table 3: CIR functions per class and corresponding gestures

Function	Class	Gesture		Function	Class	Gesture	
Select one	Main	Single tap on object		Open link	Html	Hold with one hand, double tap on link	
Select many	Main	Lasso around objects		Crop	Image	Hold with one hand, draw box with other	
Move	All	Drag object		Play/ Pause	Media	Single tap on object	
Zoom	Main, All	2 fingers move together or apart		Seek	Media	Hold with one hand, drag left or right to seek	
Rotate	All	2 fingers rotate around each other		Volume	Media	Hold with one hand, drag up or down with the other	
Duplicate	All	Double tap on object		Copy snapshot	Media	Hold with one hand, tap with other	
Annotate	All	Circle on object		Search	Main	Double tap to create search object	
Create Link	All	Drag object onto another		Open result	Search	Single tap on result to open	
Close	All	"scratch out" gesture on object		Remove result	Search	"Scratch out" gesture on result	
Select Text	Text, Html	Hold with one hand, select with other		Undo	Log	Flick left on undesired operation	
Copy	Text, Html	Hold with one hand, single tap with other		Redo	Log	Flick right on desired operation	

VI. Conclusion

This paper has presented a general architecture for a multi-touch CIR application. The functional requirements of such an application were identified. The need to incorporate a gesture processing framework was threefold: to allow for hardware abstraction, to encapsulate complex recognition algorithms and to allow for flexibility and extensibility in the gesture definitions. The GestureToolkit was chosen as a gesture processing framework and an application architecture making use of the toolkit was proposed. The process whereby gesture events are generated and passed through to the application was explained. A library of general information objects supporting the functional requirements of CIR applications was presented, allowing for four different types of information and two special objects that support search features and log features respectively.

A CIR prototype making use of the proposed framework and information object library was described. The prototype is a general CIR tool using the World Wide Web as an information space. Collaborators may search the web together to find information which can then be manipulated, shared, annotated and opened. Future work involves the evaluation of this prototype. This evaluation will provide insight into the effectiveness of the proposed architecture and the proposed multi-touch interaction techniques.

VII. Acknowledgements

The author would like to acknowledge the financial assistance of the Telkom/NMMU Centre of Excellence, the NMMU Research Capacity Development and the National Research Foundation, without which this research would not be possible.

References

[1] Hansen, Preben and Järvelin, Kalervo (2005): Collaborative information retrieval in an information-intensive domain. Information Processing and Management, 41 (5):1101-1119. Pergamon Press, Inc.

[2] Stewart, Jason, Bederson, Benjamin B. and Druin, Allison (1999): Single display groupware: a model for co-present collaboration. Proceedings of the SIGCHI conference on Human factors in Computing Systems: the CHI is the limit. Pittsburgh, Pennsylvania, United States, ACM.286-293.

[3] North, Chris, Dwyer, Tim, Lee, Bongshin, Fisher, Danyel, Isenberg, Petra, Robertson, George and Inkpen, Kori (2009): Understanding Multi-touch Manipulation for Surface Computing. In Proceedings of Interact 2009, Uppsala, Sweden.

[4] Khandkar, S. and Maurer, F. (2010): A Domain Specific Language to Define Gestures for Multi-Touch Applications. In Proceedings of the 10th SPLASH Workshop on Domain-Specific Modeling, Reno/Tahoe

[5] Hansen, P and Jarvelin, K. (2005): Collaborative Information Retrieval in an information-intensive domain. Information Processing & Management, Volume 41, Issue 5, September 2005, Pages 1101-1119

[6] Talja, Sanna (2002): Information Sharing in Academic Communities Types and levels of collaboration in information seeking and use. New Review of Information Behaviour Research, 3:143-159.

[7] Foster, Jonathan (2006): Collaborative Information Seeking and Retrieval. In Annual review of information science and technology40:329-356. B. Cronin (Ed.), Information Today, Inc.

[8] Shah, C. (2010). Coagmento - A Collaborative Information Seeking, Synthesis and Sense-Making Framework. Integrated demo at CSCW 2010. Savannah, GA: February 6-11, 2010

[9] Morris, Meredith Ringel, Huang, Anqi, Paepcke, Andreas and Winograd, Terry (2006): Cooperative gestures: multi-user gestural interactions for co-located groupware. Proceedings of the SIGCHI conference on Human Factors in computing systems. Montreal, Quebec, Canada, ACM.1201-1210.

[10] Morris, Meredith Ringel and Horvitz, Eric (2007): SearchTogether: an interface for collaborative web search. Proceedings of the 20th annual ACM symposium on User interface software and technology, October 07-10, 2007, Newport, Rhode Island, USA

[11] Twidale, Michael B., Nichols, David M., Paice, Chris D. (1997): Browsing is a collaborative process, Information Processing and Management: an International Journal, v.33 n.6, p.761-783, Nov. 1997

[12] Shah, C. (2010): Coagmento - A Collaborative Information Seeking, Synthesis and Sense-Making Framework. Integrated demo at CSCW 2010. Savannah, GA: February 6-11, 2010

[13] Golovchinsky, G., Adcock, J., Pickens, J., Qvarfordt, P., and Back, M. (2008): Cerchiamo: a collaborative exploratory search tool. In Proceedings of CSCW, 2008

[14] Ginsburg, M.; (1998): Annotate! A tool for collaborative information retrieval. Enabling Technologies: Infrastructure for Collaborative Enterprises, 1998. (WET ICE '98) Proceedings., Seventh IEEE International Workshops on , vol., no., pp.75-80, 17-19 Jun 1998

[15] Albinsson, Pär-Anders and Zhai, Shumin (2003): High Precision Touch Screen Interaction. In Proceedings of CHI'03, Ft. Lauderdale, Florida, USA, 5:105-112. April 5-10, 2003.

[16] Kaltenbrunner, M., Bovermann, T., Bencina, R., and Costanza, E. (2005): TUIO: A protocol for table-top tangible user interfaces. In Proceedings of Gesture Workshop 2005.

[17] Kammer, D., Keck, M., Freitag, G. and Wacker, M. (2010): Taxonomy and overview of multi-touch frameworks: Architecture, scope and features. In Proceedings of the EICS '10 workshop on Engineering Patterns for Multi-Touch Interfaces

[18] PQ Labs G3 Plus Product Specification (2011). http://multi-touch-screen.com/product_plus.html. [Accessed April 2011]

Ivan Sams received his BSc (Computing Sciences and Mathematical Statistics) degree in 2008 from the Nelson Mandela Metropolitan University and his BSc Honours degree the following year. He is presently studying towards his Master of Science degree at the same institution. His research interests include Human-Computer Interaction, particularly Touch Screen Interaction.