Comparing Hand-Gesture and Finger-Touch Interfacing to Navigate Bulk Image-Sequence Data

V. Du Preez, E.P.Clarkson, S. Innes, D.Q. Quach and K.A. Hawick
Computer Science, Massey University, North Shore 102-904, Auckland, New Zealand
email: {dupreezvictor, dara.quach, elliot.clarkson}@gmail.com
redmansteve2@hotmail.co.uk, k.a.hawick@massey.ac.nz
Tel: +64 9 414 0800 Fax: +64 9 441 8181

June 2013

ABSTRACT

Modern interface devices such as depth field cameras and touch sensitive screen offer new scope for navigation large and complex image data sets. Image sequences from movies or simulations can be treated as hyper-bricks of navigable data. We have implemented an image navigation interface framework that supports a compatible set of both depth-field hand gestures and touch-screen finger movements. We report on how these two apparently disparate interfaces can be combined in a unifying software architecture and explore the human computer interaction space of suitable gestural and touch idioms and metaphors suitable for rapid interactive navigation and rendering of a sequence of images from a simulation, from photographic stills, or frames of a movie. We compare the two sorts of interaction and discuss a descriptive vocabulary for these and suggest some directions for development and use in other bulk data navigation interfaces.

KEY WORDS
image data navigation; gestures; touch screen; HCI.

1 Introduction

Improvements in and cheaper costs of image capture devices are making the problem of navigating and manipulating large sequences of image data more common. Simulations too, often generate large quantities of image data. Interacting with regular hyper-bricks of data in real interactive time is quite computationally feasible using modern processing technologies if one can find the right interaction metaphors to allow a user to express appropriate navigational commands. In this paper we investigate the interaction technologies such as depth field cameras and touch screens so that users can interact with image sequence data using both gestures and multi-finger touches.

Human-Computer Interaction is a relatively mature discipline [1] and many of the key guiding principles have been well studied [2–4]. However, the widespread availability of new interface devices is leading to hitherto unexplored interaction mechanisms. The multi-touch capability of tablet computers [5] is a particularly interesting area that is still being explored by new communities of users for various disciplines.

The Kinect depth-field camera [6] is another commodity-priced device that has attracted a lot of attention as an enabler of innovative HCI modes for gaming applications [7–9], but also for applications including geospatial navigation [10], handicap and visual impairment support [11–13], and also interactive learning [14].

Widely available devices like the Kinect make possible a range of human interaction possibilities. The Kinect itself is a well integrated set of sensors [15] including cameras, orientation devices and sound capabilities. These systems and their software frameworks support detection of specific devices like paddles [16] or wands, but more interesting - and indeed natural, is for the user to use ges-
tures [17–21] to interact with a sequence or brick or images or simulation model.

Gestural interfaces are not new [22] although with the very rapid product commoditization of touch sensitive tablet computing the research and textbook literature on multi-touch and gestural systems has not yet caught up and there are surprisingly few accounts of multi-touch applications and associated experiences [23].

In this present paper we are particularly interested in enabling different devices to support image navigation, with sets of either hand gestures or screen touches having an intuitive relationship with one another. Figure 1 shows ...

HC1 experiments and applications [24] have been widely reported in the literature for tablet computing [25] on data entry [26], database interaction [27], interactive training [28, 29] and simulation interaction [30, 31]. Software development work is also reported on HCI frameworks that will further enable these applications [32].

Much recently reported research has focused on interacting with 3D objects [33, 34]. Our present paper focuses on the human user as a 3D entity [35] that must be recognized and the detected [36] human activities [37] used as feedback into any interesting running simulation. Human detection [38] requires tracking the whole skeletal body [39, 40] as well as specifics such as head and hands [41]. There appears to be a wealth of work to be done in the field of HCI in appropriately categorising and naming appropriate 3D gestures so that suitable simulation driving libraries can be formulated. As we discuss in this paper, it is not necessarily feasible or indeed desirable to solely use gestures and a hybrid approach using some mix of gestural and speech/sounds [42] may be more natural for an image sequence navigator.

Our article is structured as follows: In Section 2 we describe the client-server architecture of our software combining interpretation of Kinect gestures or touch screen interactions. We present a set of photos explaining the set of hand gestures and multi finger clicks that we developed to express image sequence navigation in Section 3. In Section 4 we discuss some of the implications for this sort of device agnostic architecture and offer some conclusions and areas for further investigation in Section 5.

2 Method

Rather than modifying existing simulations to support gesture-based input, a plug-in application was developed which would interpret gesture information sent over the network and convert this to simulated key-presses on the host operating system. Java was chosen as the language for the server for platform independent testing. To interface with the server, separate software was written for each input device to decode gesture input. This recognition software can be on the input device itself (In the case of an Android tablet), on the host machine (in the case of the Kinect), or anywhere on the reachable network, provided it has fast access to the raw data provided by the input device. When gestures are recognised, a short string describing the detected gesture is sent over the network to the server application. These loosely coupled systems communicating over a network allows existing simulation software to quickly and easily make use of gesture input, often without any modification to the source code.

While this approach is flexible and easy to both implement and use, it is limited in that it can only offer as much control as the target program provides with hot-keys. This means that gestures such as rotation must be done iteratively; simulations will not be able to respond in real-time to the rotational degree of the gesture without modifications to the target simulation.

2.1 Input Device - Android

The Android API has large support for gestures, it being the main way to interact with most Android devices, and was very easy to implement. Gesture listeners, which cater for most generic gesture inputs, have been a part of the API since its creation. There is still some modification needed in order to recreate more specific gestures, such as fling left and fling right, but these were easy to build on top of the provided interface methods which cover a lot of the more common gestures.
Figure 3: Gesture to Key-press server, ready to turn gestures from the network into key-presses on the host machine.

**Swipe Left** Wait for a touch event to be placed and record the position. If the finger is raised in a different position on the screen with a lesser x dimension than the down even was recorded having then a swipe left has occurred.

**Swipe Right** Wait for a touch event to be placed and record the position. If the finger is raised in a different position on the screen with a greater x dimension than the down even was recorded having then a swipe right has occurred.

**Pinch** When 2 fingers are placed on the screen their positions and the distance between them is recorded. When the fingers are lifted off of the screen take the positions of each and find their distance apart from each other. If the second recorded distance is less than the first it is a pinch gesture

**Spread** When 2 fingers are placed on the screen their positions and the distance between them is recorded. When the fingers are lifted off of the screen take the positions of each and find their distance apart from each other. If the second recorded distance is greater than the first it is a spread gesture

**Touch** If a touch down event is recorded followed very quickly by an off touch event with very similar positions then a touch gesture has occurred.

**Long Touch** If a touch down event is recorded followed by an off touch event after 2 seconds and these two events have similar recorded positions then a long touch gesture has occurred.

### 2.2 Input Device - Kinect

The Kinect is made for tracking general body movements rather than fine manipulators. This restricts the variety of gestures the device is capable of accurately reporting. It is important to keep this in mind when designing gestures to be recognised by this device, using broad and deliberate motions to account for the lack of accuracy. We used the skeletal tracking to invoke these gestures using joints that represent major points in the skeleton body. This allowed us to track regions in the recognition with some positioning using x and y coordinates.

We achieved a gesture vocabulary that does not detect unwanted gestures. This makes it difficult to implement gestures that are consistent and reliable. We therefore implemented a rule-set that subtlety detects the required gesture. Figure 4 show how the implemented zoom-in gesture, similar to Android’s pinch-to-zoom gesture is performed by bringing both hands above the neck and spreading them apart at a greater distance than the users shoulder width. Once the gesture has been performed the software idles until a new gesture is recognised.

### 3 Experimental Simulation Results

The system was tested on three applications, with two separate input devices. The server was run on a Mac host machine, and loaded with configurations of hot-keys for VLC media player, Xee image viewer, and Animaux, an agent based simulator running the Ising model. Figure 6 displays the gestures used and how they were mapped to
Table 1: Comparing gestures between the Kinect and the Android tablet.

<table>
<thead>
<tr>
<th>Kinect Gesture</th>
<th>Android Gesture</th>
<th>VLC</th>
<th>Xee</th>
<th>Ising Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swipe Left</td>
<td>Medium Forwards Jump</td>
<td>Next Image</td>
<td>Increase Critical Coupling</td>
<td></td>
</tr>
<tr>
<td>Swipe Right</td>
<td>Medium Backwards Jump</td>
<td>Previous Image</td>
<td>Decrease Critical Coupling</td>
<td></td>
</tr>
<tr>
<td>Pinch</td>
<td>-</td>
<td>Zoom Out</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Spread</td>
<td>-</td>
<td>Zoom In</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Hand Raise</td>
<td>Tap</td>
<td>Pause/Play</td>
<td>-</td>
<td>Next Iteration</td>
</tr>
<tr>
<td>Long Hand Raise</td>
<td>Long Tap</td>
<td>Stop</td>
<td>-</td>
<td>Run Simulation</td>
</tr>
</tbody>
</table>

Figure 6: Mapping gestures to actions in programs.

3.1 Simulations Software

The system exhibited positive results when tested with advanced agent-based simulation software running the Ising model, making it appropriate for displaying the model to an audience or for allowing observers to interact with the model in an intuitive way. The existing software interface had a number of problems. It was crowded with intimidating elements, and the key combinations were model specific and occasionally esoteric. In comparison, the gesture overlay remained simple to operate provided it was configured correctly, completely bypassing the traditional methods of control. Configurations specific for particular models could be loaded and mapped to familiar gestures that reflect the action taken on the model.

A user’s existing knowledge can be transferred from even non-scientific applications. Knowing that performing a ‘hand raise’ gesture on the Kinect triggered a pause/play command in a media player can be directly applied to the agent-based simulation model, with the implication that the same gesture in the agent-based simulator would result in some kind of temporal manipulation (a ‘next iteration’ command for the Ising model.)

It was even possible to define a gesture to switch configurations on the gesture server, making it easier still to interact with complex models. Although introducing modes into an already complex system would result in a poor user experience.
experience, [43] it makes sense to switch configuration to match the model displayed in the simulation software if it changes. This allows users to perform the same familiar set of gestures and have it map to the control configuration suited for that model.

Hiding the interface and providing a standard set of interaction gestures means that non-scientists can interact with and manipulate models without having to have a deep understanding of the parameters that modify their behavior. This encourages exploration and investigation into what would otherwise be an intimidating concept to someone unfamiliar with complex systems.

3.2 Input Devices

While the two input devices tested are able to share many concepts such as the same motion to execute gestures (Further discussed in section 4.1), they also differ significantly in other areas. It is quite simple for an Android device to detect the start and end of a gesture, as these events correlate with the introduction and removal of fingers from the touch-screen. The Kinect is harder to work with in this regard, as the points of interest for gestures are almost always visible. The requirement for a voice command to initiate a gesture was briefly considered, but this was ultimately rejected in favour of comparing the location of interest points in relation to others. For example, a ‘swipe left’ gesture would only start when the left hand was below the waist, and the right hand between the waist and shoulders, and further away from the body than the right shoulder. This means transition from one device to the other is a more natural process, as there are no additional steps required; If the user is relaxed and acting in a sensible manner, a gesture on one device occupies the same logical space as on another. Gestures become more device-agnostic and universal, which means less to remember for the user.

An interesting observation is that both devices tested have some kind of processing capability. The Kinect outputs skeletal data which requires only minimal processing to convert into a useful gesture, and Android devices are completely capable of doing all gesture processing themselves and simply passing along a message indicating a gesture has been detected to the server. This may not always be the case, for example if the host machine is engaged in processing gestures from a web-cam. However, when it is the case, this lightens the load on the host machine, meaning its CPU resources are free to run more intensive simulation software. This could be important in interfacing with real-time simulations.

4 Discussion

Gestural computer interfacing is still in a relatively early stage of development and there is great scope for identifying and naming standard gestures for more widespread use in applications. Some of the touch and gestural notions like “pinch to zoom” have now become widely accepted and understood. Other named intuitive gestures will likely emerge from ongoing work in this area.

4.1 Homomorphic Gestures

Some gestures have a very clear method of execution across various devices. A 'pinch' gesture, for example, can be intuitively executed on the Kinect by bringing both hands together. The same gesture can be executed on Android by pinching the fingers together on the touch surface. These gestures can be considered homomorphic. Homomorphism is a desirable trait because such gestures facilitate information transfer - Skills learned on one platform become obvious and intuitive on another. The system architecture supports this paradigm by easily linking these gestures to the same action on the host machine, easily bringing together the motion with the desired intent.

4.2 Representing Continuous Data

Some gestures are well suited for delivering continuous output - That is, a continuously changing stream of data, which could be interpreted from the height of a hand, or the distance between fingers, for example. This is desirable because the rich data it provides would be suitable for modifying parameters of simulations in real time. However, there are some difficulties in transparently conveying the information to a simulation application. The implemented hot-key-based approach is only really suitable for discrete information packets, and a continuous stream would have to be represented as a plethora of key-presses. Something like a constantly changing floating point value is difficult to represent at all with this system.

In order for a program to receive continuous information in an input device agnostic manner, the program could listen for information on a stream input such as the stdin file descriptor, which could be provided either by a server listening for streams of gesture data over the network, or being piped in from a program which talks directly with an input device, bypassing the network entirely.

A potential problem with continuous data gestures is the requirement for some kind of termination of the gesture. For example, an intuitive end of a rotation gesture using a touch device is to simply remove one hand form the touch surface. This stops the continuous stream of data, and the final value of any variable bound to that data can be set as the last value received. However, for the Kinect, the situation is slightly less obvious - There is no clear way to abruptly end a gesture such as hand elevation. Lower-
ing ones hand out of the active gesture area is impossible without unwanted modification of gesture data. To remedy this, a further signal must be given to the device without modification of the interest point the Kinect is monitoring. This could be done with a voice command, or the movement of another, non-tracked interest point. However, this introduces the problem that performing this gesture is quite different between devices, which could make the process less intuitive.

5 Conclusions

We implemented gestures on the Kinect based of conventional touch-device gestures, we saw how these gestures relate and if there are some commonalities. We have shown that the Kinect could be used in the same manner as you would a touch-device, using only hand based gestures. The software developed shows that this could replace conventional key-bindings and change the way humans interact with computers. Hardware such as the Kinect shows that this could easily be achieved.

This will allow users to interact with computer applications as they would a touch-device. However the technology has some limitations, such as no finger recognition which restricts the amount of gestures that can be implemented. These gestures work very well with all the applications as a navigation tool for streams of data in picture and video software. The Kinect works well for the implemented gestures and shown very accurate and reliable data however only a small number of gestures could be implemented due to restrictions within the device.

Considering there is no set gesture vocabulary describing how humans should interact with these applications, we believe the gestures implemented show how ease of use is similar to accustomed finger gestures. Since these finger gestures such pinch to zoom are conventional for all touch devices these gestures we implemented should have the same effect with HCI.

We believe these gestures will become more generalised for humans to interact with a more similar technology. This will likely be available in standard user applications. We have shown that these uses could support video, imaging and simulation applications as a manipulation tool. The implemented gesture methods mechanics that could be much deeper refined, allowing scope to investigate a higher hierarchy of gestures using different devices and more complex gestures.

New technology such as the Leap Motion and Kinect 2.0 could allow whole new spectrum of gestures for human computer interaction. Increased precision and performance in finger recognition should allow these gestures used in touch devices to be more generalised.

References


