A Symmetric Interaction Model for Bimanual Input

by

Celine Latulipe

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2006

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

People use both their hands together cooperatively in many everyday activities. The modern computer interface fails to take advantage of this basic human ability, with the exception of the keyboard. However, the keyboard is limited in that it does not afford continuous spatial input. The computer mouse is perfectly suited for the point and click tasks that are the major method of manipulation within graphical user interfaces, but standard computers have a single mouse. A single mouse does not afford spatial coordination between the two hands within the graphical user interface. Although the advent of the Universal Serial Bus has made it possible to easily plug in many peripheral devices, including a second mouse, modern operating systems work on the assumption of a single spatial input stream. Thus, if a second mouse is plugged into a Macintosh computer, a Windows computer or a UNIX computer, the two mice control the same cursor.

Previous work in two-handed or bimanual interaction techniques has often followed the asymmetric interaction guidelines set out by Yves Guiard's Kinematic Chain Model. In asymmetric interaction, the hands are assigned different tasks, based on hand dominance. I show that there is an interesting class of desktop user interface tasks which can be classified as symmetric. A symmetric task is one in which the two hands contribute equally to the completion of a unified task. I show that dual-mouse symmetric interaction techniques outperform traditional single-mouse techniques as well as dual-mouse asymmetric techniques for these symmetric tasks. I also show that users prefer the symmetric interaction techniques for these naturally symmetric tasks.

Acknowledgements

Collaborators

There are four people who deserve recognition for technical contributions to the work contained in this thesis. Kevin Moule assisted with the initial development of the symPut driver under Linux, and acted as a technical resource for the Linux/Xlib/OpenGL development that followed. Stephen Mann collaborated with me on the symSpline technique, providing algorithms to make the splines behave appropriately in response to the interaction that I designed. Ian Bell was a partner in the development of the symTRC and the ToneZone techniques. Ian contributed his extensive knowledge of colour and tone manipulation and general image processing algorithms to the development of the symTone application. Sumair Ur Rahman ported both the symPut driver and the symDraw application to the Macintosh platform and provided assistance as I ported symTone to the Macintosh.

Previous Publications

Many of the results presented in this thesis have been published in papers in various conferences:

- The simultaneous rotation and translation experiment described in Chapter 4 was published as a short paper in British HCI 2005 [LKC05a].
- The image registration experiment described in Chapter 4 was published as a full paper in UIST 2005 [LKC05b].
- The symSpline technique and experiment described in Chapter 5 was published as a full paper in CHI 2006 [LMKC06].
- The symTRC technique and experiment described in Chapter 6 was published as a full paper in GI 2006 [LBCK06].

I thank the anonymous reviewers of all of these papers, who provided valuable feedback.

Funding

This work was made possible, in part, by the provision of Ontario Graduate Studies (OGS) scholarships. This work was indirectly supported through my supervisors' NSERC research grants.

Personal Acknowledgements

A number of people in the Computer Graphics Lab have been supportive and encouraging and have also acted as pilot subjects for many of my experiments. Selina Siu, Patrick Gilhuly, Stefanus Dutoit, Erin Lester and Gilad Israeli were all especially helpful. Rob Kroeger deseves special thanks for wise advice and support as well as for introducing me to the Computer Graphics Lab in the first place. Bill Cowan was my supervisor when this research began and deserves thanks for his encouragement and support in the initial stages of this research. Other people in the School of Computer Science who have been encouraging and supportive include Michael Terry, Edward Lank, Sandy Graham and all of the staff in the Computer Science Office, especially Wendy Rush and Gail Chopiak.

I owe much to the guidance, support, advice and encouragement of my supervisors, Charlie Clarke and Craig Kaplan. Charlie and Craig have both been truly enthusiastic about my research, and their excitement has helped to get me through the tough spots.

A work such as this cannot be completed without the love, support and encouragement of family and friends. I would especially like to thank Pam Charbonneau, Melanie Adrian and Rudy Broers. I also thank my parents, John Latulipe and Huguette Miller and my uncle, Richard Conrad. Thanks to Ron Singh, my brother-in-law, for hardware contributions and to all of the Singh family for their support.

Finally, I want to thank my husband, Rob Singh-Latulipe, and my children, Colin and Abby, for allowing me the time and energy needed to complete this thesis. Without Rob's emotional and financial support, I would not have started or finished this thesis.

Dedication

I dedicate this thesis to my mother, Ruth Elizabeth Conrad (1939-2004). She was a vice-president and a wonderful mother at the same time, and she still inspires me. I also dedicate this thesis to all 'Career Moms', who constantly have to balance their career goals with their family goals. It's not easy, but it's worth it to show our daughters that they can accomplish whatever they set their minds to.

Trademarks

Windows is a trademark of Microsoft Inc.

Carbon and MacOS X are trademarks of Apple Computer Inc.

UNIX is a trademark of the X/Open group.

Photoshop is a trademark of Adobe Inc.

Smart Board is a trademark of Smart Technologies Inc.

Paint Shop Pro is a trademark of Corel Inc.

Lemur is a trademark of JazzMutant Inc.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis describes a model for symmetric, bimanual interaction in the human-computer interface. Bimanual interaction is interaction involving both hands. *Symmetric* bimanual interaction is interaction in which both hands contribute equally to the completion of the task. This thesis describes implementations and evaluations of symmetric interaction techniques.

While there are many tasks in our everyday lives that require the cooperative use of both hands (steering a bicycle, tying shoelaces, and clapping hands, to name just a few), there are few tasks that can be performed in current computer interfaces with both our hands contributing equally. Given the prevalence of real-life activities in which we can use both hands together, it seems odd that computers don't allow this type of interaction. Typing text using a keyboard is the one notable exception. The limitation in current systems is two-fold: first, most computers only have a single spatial input device, and second, most commercial software, including operating systems and window managers, are written based on the paradigm of a single pointing device and a single cursor.

Some obvious questions arise: can these limitations be removed? And, if these limitations were removed, could anything useful or interesting be done on a computer with two pointing devices and two cursors?

The answer to the first question is yes. Typical desktop computers offer a single pointing device. When a second pointing device is plugged in, the spatial input streams from the two devices are combined into a single spatial input stream, causing the two devices to fight for control of a single cursor. The symmetric input driver described in this thesis, symPut, separates the spatial input from two mice into individual input streams that can be used to control separate cursors.

The answer to the second question, whether anything useful or interesting can be done on a computer with two pointing devices and two cursors, is also yes. This thesis will present a

number of applications and techniques which show that useful, creative work can be performed more quickly and with higher user satisfaction when a second pointing device and a second cursor are present.

1.1 Symmetric vs. Asymmetric Interaction

Symmetric interaction is interaction in which the two hands contribute to a task equally. In the Human-Computer Interaction (HCI) literature and the Psychology literature, tasks that require the use of two hands are termed 'bimanual'. An alternative type of bimanual interaction is *asymmetric* interaction, where different sub-tasks are performed by the dominant and non-dominant hands.

Consider the act of using modifier keys, such as holding down the shift key for the selection of multiple icons or files. The modifier key is pressed down by the non-dominant hand and then the objects are selected by the cursor controlled by the dominant hand. This is a useful skill that people master without much thought. This skill is characterized as an asymmetric bimanual task and conforms to the principles of Guiard's Kinematic Chain model [Gui87]. In asymmetric interaction, the non-dominant hand sets the frame of reference in which the dominant hand works, and the dominant hand does work at a finer level of spatial and temporal detail. Generally, the non-dominant hand plays a supporting role. Handwriting is the standard example of asymmetric interaction: the non-dominant hand positions the paper and holds it in place and then the dominant hand does the writing. Asymmetric interaction has been well-investigated within the HCI community [Gui87, BH99, HCS98, HPP+97]. Guiard's model has been used in developing two-handed computer interfaces and will be described in detail in Chapter 2. In particular, a number of graphical 3-D interfaces have been developed in which the dominant hand does most of the work, while the non-dominant hand handles macro-level details such as view manipulation [KBS94, KFBB97, BSP+93, RR96].

1.2 The Need for Symmetric Interaction Research

The symmetry to which I refer in this research is not a mathematical symmetry in which both hands perform the same movements as if a mirror was drawn down the middle of the body. The asymmetry that leads to Guiard's Kinematic Chain model is an asymmetry in both the temporal and spatial level of detail of the work performed by each hand. Thus, in tasks that fit Guiard's model, the non-dominant hand is not required to move as frequently, nor does the hand make movements that require fine spatial dexterity. I define a symmetric task as one in which both hands perform work at comparable levels of spatial and temporal detail.

There are very few successful symmetric interfaces. The Twister and Bender techniques described in Chapter 2 are two examples from the HCI literature. However, real life is replete with examples of symmetric interaction: playing the clarinet, skipping rope, shuffling cards [1], folding linens, drumming and rolling out pie dough, are just a few examples of activities in which both hands play important and seemingly symmetric parts. What do these activities have in common?

- Both hands contribute simultaneously to the activity, and at a similar level of detail.
- Neither hand leads or guides the other.
- The contributions of the two hands must be precisely timed.

Are there activities in the computer interface that could be performed symmetrically, similar to the way the activities above are performed? This is a very difficult question to answer, because the computer interface has evolved *without* the ability for the two hands to contribute equally. Thus, tasks which might be naturally bimanual and symmetric have been artificially broken down into sub-tasks that can be performed asymmetrically or in serial by a single hand. Because users have been performing tasks in this broken-down, artificial manner for so long, it is hard for people to imagine performing them in any other way.

The positioning and sizing of a typical application window is a perfect example. If I had a real object that could be expanded and contracted (like an accordion) and I wanted to move it from position A in contracted form to position B in expanded form, I would use both of my hands to move it and I would stretch it out into the expanded form *while* I was moving it. With an application window, a user must acquire the application title bar and drag it to reposition the window. Then a separate operation is required to resize the window: the user must acquire the window corner and manipulate it to resize the window. Often, after the window has been adjusted to the preferred size, it must then be moved again, because it is not quite where the user desires it to be. The user is forced to switch repeatedly between the sizing and the positioning of the window. If the user had two separately controlled cursors, she would be able to grab two opposing corners of the window and maneuver the window into its desired position and size in a single, two-handed gesture.

In "The Role of Kinesthetic Reference Frames in Two-Handed Input Performance", Balakrishnan and Hinckley state:

> In other tasks, such as two-handed map manipulation, the gesture of bringing one's hands together is an important skill. Such movements seem characteristic of

[1] "The riffle shuffle (sometimes called a faro shuffle) is the traditional method of shuffling. The deck is divided approximately in half, one part going into the left hand and the other part going into the right hand. Then by riffling the thumbs over the edges of the cards, the two halves of the deck are interleaved together."[Loy]

> *symmetric* bimanual action, as opposed to the asymmetric action we have focused on so far. This raises the issue of whether symmetric and asymmetric bimanual gestures are fundamentally different, or whether a generalization of Guiard's principles could apply to both cases... With the exception of Leganchuk et al., there is little experimental data on symmetric two-handed input tasks, suggesting that this is an area ripe for further experimental study. [BH99]

This thesis is a direct answer to the proposal that symmetric two-handed input is an area ripe for further experimental study.

1.3 Hypothesis

The hypothesis investigated in this thesis is that there exists an interesting class of tasks in the human-computer interface that are naturally symmetric and, for this class of tasks, the use of symmetric interaction techniques will yield higher performance and increased user satisfaction. Investigating this hypothesis requires the development of a symmetric interaction model to classify the symmetry of different tasks, the development of symmetric interaction techniques, and the formal evaluation of those techniques.

1.4 The Symmetric Interaction Model

The Symmetric Interaction Model that is described in Chapter 3 includes a *taxonomy of symmetries* that can be present in a user interface. Because symmetry is an over-loaded term, this taxonomy serves to define the different types of symmetry that will be relevant to this research. Thus, the taxonomy of symmetries serves as a nomenclature reference for the rest of the thesis.

The Symmetric Interaction Model also details what constitutes symmetric interaction in everyday tasks. A *Symmetry Spectrum* is provided that shows how some typical real-life tasks can be categorized. This is then expanded with an axis for the skill-level of the task into a *Symmetry-Skill Plane*. The Symmetry-Skill Plane is also discussed in the context of everyday task examples. An example of a Symmetry-Skill Plane applied to a specific domain (that of piano music) is included.

The Symmetric Interaction Model gives *guidelines* for the implementation of symmetric interaction techniques in the human-computer interface. These guidelines were followed in the creation of the symmetric techniques that are described in Chapters 4, 5, and 6.

The final contribution of the Symmetric Interaction Model is a set of *rules of thumb* for interaction and interface designers. The rules were derived from my experiences in developing

and testing symmetric and asymmetric interaction techniques. These rules can help a designer who is considering a bimanual technique for a given task to determine whether the task should be designed with a symmetric interaction technique or an asymmetric interaction technique.

1.5 Platforms and Devices

It is my long-term goal to see dual spatial input and dual cursors as a standard feature of desktop computers. While that goal is not meant to serve as a measure of success in the immediate future, it has guided the choice of platforms and devices for this research.

Although the most popular and widely used platform is the Microsoft Windows PC, current versions of Windows do not make it easy to access devices and separate out spatial input from those devices. Each major release of Windows changes the model and APIs for accessing low-level device information. For this reason, I initially chose Linux as my implementation platform. The symPut driver and the applications built on top of it have also been ported to the Macintosh platform.

A symmetric interaction technique in a 2D application requires two spatial input devices that each provide x and y coordinate information and buttons for accessing program functionality. There is obviously a tight coupling between the way an application can be designed and the type of input devices an application can handle. If a technique requires six different buttons, then a special device with six different buttons will be required. While there are merits to using devices that are specialized for the task at hand, there are also drawbacks, mainly in the general applicability of the research and the interaction techniques to other tasks.

Much research has been done in the HCI community in the area of two-handed interaction and this research will be described in Chapter 2. Most of the implementations described in previous studies of two-handed interaction have used specialized devices: digital pens, pucks, digitizing tablets, flocking devices, etc. While most of these studies have clearly shown the benefits of two-handed interaction, the standard computer desktop still has only one pointing device: a mouse.

The research that I have done for this thesis has used a simple setup consisting of a standard computer with two standard Universal Serial Bus (USB) mice. It is inexpensive for a user to add a second mouse to a computer system and almost all computers have at least two USB ports. [2] Additionally, there is no learning curve associated with mice, as there would be with specialized devices: users are already comfortable using a computer mouse. I believe that by using two standard mice for my research on symmetric interaction, I am providing evidence to the computing community that two-handed interaction is useful, attractive and affordable for a general audience rather than just a specialized setup useful for industrial designers.

[2] Computers that have only one available USB port can use an inexpensive USB hub to add more USB ports.

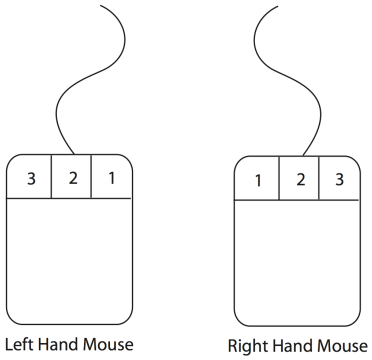


Figure 1.1: Mouse button assignment used by the symPut driver. Button 1 is the inner button on both mice, Button 2 is the middle button on both mice and Button 3 is the outer button on both mice.

1.6 The symPut Driver

In order to implement and study symmetric interaction techniques, it was necessary to create a low-level driver that would separate input streams from two computer mice such that they could direct independent cursors. The driver requires the use of two standard, 3-button USB wheel mice. The mice can be either traditional ball mice, or optical mice. The mice can also be wired or wireless.

The driver is specially designed to afford symmetric interaction techniques and the development of applications that encompass them. Applications that use the symPut driver can easily handle symmetric interaction techniques that require activation of the same button on both devices. The buttons are assigned with reflective symmetry, as shown in Figure 1.1. The left button on the right mouse and the right button on the left mouse both activate *main* mouse button events. Similarly, the right button on the right mouse and the left button on the left mouse both activate *outer* mouse button events. This symmetric assignment of button functionality is based on research findings in Psychology and will be discussed in detail in Chapter 3.

1.7 Applications and Techniques

Two different software applications, which include a variety of different symmetric interaction techniques, have been developed as a part of this research. Both of these applications are built using the symPut driver and operate on the Linux and Macintosh platforms.

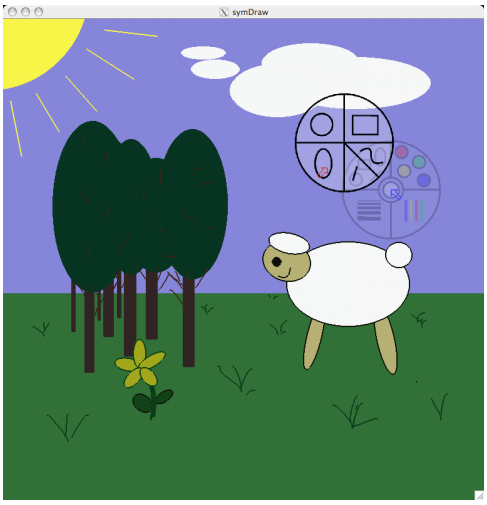


Figure 1.2: A sample screenshot from the symDraw application with the popup menu displayed.

The first application, symDraw, has been a testbed application used for experimenting with different symmetric interaction techniques. It is a 2-D vector-based drawing program with a completely symmetric interface presentation. symDraw includes techniques for symmetrically creating shapes as well as symmetrically rotating, scaling, and translating shapes. symDraw also includes a symmetric technique for the manipulation of spline curves. Figure 1.2 shows a sample image created with the symDraw program, as well as the popup menus that are used to access program functionality.

The second application, symTone, is an image processing application that was designed to allow novice users to explore tone editing of their digital photographs in an easy and intuitive manner. symTone includes a symmetric technique for manipulating a tone-reproduction curve. It also includes a symmetric technique for controlling the input and output tone ranges of a digital image. Figure 1.3 shows a sample screen shot of the symTone application, with the tone-reproduction curve being edited with two cursors. The striking feature of the symTone techniques is that symmetric interaction is used to manipulate non-spatial data (the tone levels in the image) using a spatial manipulation paradigm (the tone reproduction curve).

Most of the techniques described in this thesis have been evaluated using controlled laboratory experiments. This includes the symmetric technique for rotating and translating objects and the symmetric technique for rotating, translating and scaling objects. These techniques were previously described by Kurtenbach et al. as 'two-handed stretchies' [KFBB97], but were never evaluated. The symSpline technique and the symmetric tone-reproduction technique have also been formally evaluated in the laboratory setting. These applications and techniques support the hypothesis by demonstrating the existence of interesting and useful tasks

Figure 1.3: A sample screenshot from the symTone application

that are naturally symmetric and that benefit from the application of symmetric interaction techniques.

1.8 Objectives

I had a number of goals that directed the development of the Symmetric Interaction Model. These goals helped to form the model and helped to drive the development of the symPut driver and the techniques and applications that followed.

1.8.1 Multiple Cursors

Asymmetric interaction techniques in HCI studies tend to have a single cursor controlled by the dominant hand while the non-dominant hand controls some macro functionality such as panning or zooming or menu location. For this type of supportive role, a second cursor is not necessary. However, for symmetric interaction, it is necessary for the two hands to have equal affordances for interacting with widgets and data, and this requires two cursors.

There are issues that need to be addressed if multiple cursors are included in an application. There must be a way for users to differentiate which mouse controls which cursor. There may also be issues for users if the cursors cross each other horizontally (such that the cursor controlled by the left mouse is to the right of the cursor controlled by the right mouse). The Symmetric Interaction Model described in Chapter 3 addresses these issues.

1.8.2 Accessibility

Current two-handed interaction systems use specialized roles for each hand, usually following the principles set out by Guiard. These systems tend to deal with different hand preferences among users in a simplistic manner. Typically, the user has to move the input device for the dominant hand to the opposite side of the keyboard and change software settings so that the buttons and other inputs get mapped appropriately for his or her particular handedness. This puts the onus on the user, especially left-handed users, to do work to make the system user-friendly.

In "The Two-Handed Desktop Interface: Are we there yet?", MacKenzie and Guiard argue that the current desktop setup does not easily afford two-handed interaction that conforms to users' hand-dominance preference [MG01]. They argue that the desktop overloads the right hand, which is bad for right-handed people, because many of the functions could easily be deemed 'supporting' and performed by the left hand. They argue that the situation is slightly better for left-handed users, but there are still obstructions to efficient manipulation and performance of common tasks. One of the areas they see for easy improvement is in scrolling, which they argue should be done by the non-dominant hand. They suggests a scroll wheel or touch strip be added to the sides of the keyboards so that scrolling can be done by the non-dominant hand.

My objective in developing the symmetric interaction model is to eliminate system-designated hand specialization. I want to be able to create applications and techniques that are equally accessible to left- and right-handed users, as well as to those who are ambidextrous, without forcing individual users to configure the system (at the hardware or software level) for their particular handedness. The dual-mouse interface that is proposed for a variety of applications throughout this thesis answers that demand because each mouse can be used individually for single-handed interactions, without requiring a user to adjust settings based on handedness. It is interesting to note that this setup provides a solution for the scrolling suggestion of Mackenzie and Guiard. Since each mouse has a scroll wheel, scrolling can always be performed by the non-dominant hand. This is just one example of how a dual-mouse desktop eliminates handedness biases and allows for richer interaction for all users, whether that interaction is symmetric or asymmetric or requires only one hand.

1.8.3 Skill Affordances

Finally, a potential benefit from a symmetric interface design is the affordance for users to develop mousing skill in their non-dominant hand. Because users have the opportunity to perform any one-handed task (such as a single click to acquire an object) using either hand, users can make a conscious effort to use their non-dominant hand on a regular basis and therefore build up mousing skill in their non-dominant hand. There are a number of reasons

that a user might want to do this. First, users may realize the benefits afforded by the parallelism inherent in symmetric interaction and decide that to build up mousing skill in their non-dominant hand will help them to work more efficiently. Second, users of traditional single input systems (such as the desktop computer with one mouse) or asymmetric two-handed systems (such as a trackball in the non-dominant hand and a stylus or mouse in the dominant hand) may end up suffering from repetitive stress injuries such as carpal tunnel syndrome. For these users, the ability to evenly share the work-load across the two hands may be beneficial.[3]

Although users may decide to make a conscious effort to use their non-dominant hand for one-handed tasks such as clicking, there is also the possibility of users developing skill in their dominant hand without making any conscious effort to do so. If an application is designed such that a primary task in the application *requires* the use of both hands, a user of that application will develop skill in the non-dominant hand automatically through application use. This unconscious skill development is highly desirable, because it comes without cost to the user. Thus, it is an objective that the symmetric interaction model afford users the ability to develop skill in their non-dominant hand without any conscious effort.

1.9 Organization

The remainder of this thesis presents the Symmetric Interaction Model and the techniques and applications that have been designed and implemented based on this model. A number of these techniques have been quantitatively and qualitatively evaluated in controlled lab experiments. These experiments and the results are also described in detail.

Chapter 2 will present the previous work in the area of two-handed interaction. That work will be divided between previous work in asymmetric interaction, including Guiard's model, and previous work in symmetric interaction. The previous work that is related to the implemented techniques in Chapters 4, 5, and 6 will be presented in those chapters.

Chapter 3 presents the Symmetric Interaction Model, including the taxonomy of symmetries, the symmetry spectrum, the symmetry-skill space and the guidelines for implementing symmetric interaction techniques.

Chapters 4, 5 and 6 present the applications and techniques developed based on the Symmetric Interaction Model. In Chapter 4, the symDraw application is described. This is followed by evaluations of the symmetric techniques for simultaneous rotation and translation of objects and for simultaneous rotation, translation and scale of objects. In Chapter 5, the symSpline technique is described, along with the experimental evaluation of that technique. Chapter 6

[3] I know of at least two right-handed people who have switched to mousing with their left hand in order to give their right hands and wrists a chance to recover from repetitive stress injuries. Both these users reported that they were able to use their mouse quite well with their left hands after just a short period of practice.

includes a description of symTone, the digital image processing application. Chapter 6 includes an experimental evaluation of symTone's symmetric tone-reproduction curve technique.

Chapter 7 presents conclusions, including a summary of the contributions of this work. This is followed by a discussion of potential directions for future work in Chapter 8.

Chapter 2

Previous Work

Chapter 1 introduced the main idea of this thesis: that naturally symmetric user interface tasks exist and symmetric interaction techniques will yield better performance on those tasks. This idea is rooted in research that others have done in related areas, but differs from that research in significant ways. In this chapter, the relevant previous work is described, first for categorizing input devices and modeling interaction. Then work related to bimanual interaction in general is presented, followed by previous work in the area of symmetric interaction. In addition some related psychology work is described, as well as some related systems work dealing with multi-input device implementations. This chapter ends with a summary of how the research presented in this thesis differs from the previous work described.

2.1 Modeling Devices and Interaction

A number of researchers have investigated and described models and taxonomies related to input devices and graphical interaction techniques. These are described here as background for the Symmetric Interaction Model presented in Chapter 3.

Foley et al. presented a taxonomy of interaction techniques in 1984 demonstrating that more than one technique can be used for a given task [FWC84]. However, the characteristics of the device used to perform the task were not part of the taxonomy. Buxton addressed this limitation by modeling continuous-input devices, paying attention to the feel of the devices for the user and the number of dimensions of input the device could generate [Bux87].

In 1986, Buxton presented the concepts of 'chunking and phrasing' [Bux86]. The main idea in this work is that interaction gestures that correspond to single words or phrases in the user's vocabulary are more meaningful, easier to use and acquire, and more compatible with the user's model of how things should work. This concept is important, but has been difficult

for user interface designers to follow because of the limitations of spatial input in standard systems.

Card, Mackinlay and Robertson presented a taxonomy of input devices based on primitive and compound operations of the devices [CMR90]. The same authors followed up on this taxonomy with an analysis of input devices using a parametrically described design space [CMR91]. Their work took into account both the device footprint and the device performance in standard Fitts's law tests to determine which types of devices might be more effective than the standard mouse.

A three-state model of graphical input was presented by Buxton in 1990 [Bux90]. The three-state model was developed to help create a vocabulary and understanding of the relationship between input devices and the techniques that can be used with those devices. The three-state model can be applied to single-button or multi-button mice, touch tablets, touch screens and other devices. It is useful for determining the range of abilities of each of these devices, which in turn can be used to determine the techniques the device affords.

In 1990, Bleser presented a tool that assisted with selection of interaction technique and input device based on a device taxonomy that explicitly incorporated task requirements, the physical attributes of the device and the separability of input degrees of freedom [BS90]. Jacob et al. also showed the necessity for matching the input device to the task in their work on the integrality and separability of input devices [JSMMPM94]. They advocate looking beyond the mechanical affordances of the input devices and considering the relationship between the perceptual structure of the task and the controls afforded by the device. They hypothesized that performance would be improved if the perceptual structure of the task matched the control affordances of the devices and showed this to be true comparing two different devices and two different tasks.

These models and taxonomies have focused on the use of a single spatial pointing device and unimanual interaction. While these models have certainly helped to increase the understanding of different types of devices and different ways of modeling single-input interaction techniques, there is no straightforward application of these models to bimanual interaction. In the next section, the leading model of bimanual interaction is introduced and described. The input device taxonomies that have been described can help designers determine which device should be applied to a task and this can be useful for designing asymmetric techniques in which the hands play different roles. For symmetric tasks, the hands should use identical devices and other factors related to symmetric interaction are likely to be more important than specific device characteristics.

2.2 Bimanual Interaction in HCI

Two-handed, or bimanual, interaction has received much attention in HCI research. Bimanual interaction can be asymmetric, such as in handwriting, where one hand is doing the detailed work of writing and the other hand plays a supporting role of holding the paper in place. Bimanual interaction can also be symmetric, such as steering a bicycle, where both hands are performing work at similar levels of detail. Bimanual interaction has promising advantages over unimanual interaction. Before describing some of the results of previous work in this area, it is necessary to define the differences between symmetric and parallel performance. Parallelism is an important metric by which the success of a two-handed interface can be measured. Parallelism has the benefit of being a metric that is easily defined and measured: it is the percentage of time in which the two hands are moving simultaneously during the completion of a task.

2.2.1 Symmetric versus Parallel Performance

Although related in a subtle way, there is a difference between symmetric and parallel performance. If the two hands are cooperating in parallel, they are both working at the same time. If two hands are cooperating symmetrically, that means both hands are working at the same skill level (at the same levels of temporal and spatial detail). There are four cases to consider:

- Performance is symmetric and parallel. In this case the two hands are engaged in a task where both are doing work of similar temporal and spatial detail and the two hands are working simultaneously. Real life examples include steering a bike, rolling out pie pastry, clapping hands, juggling and carrying a large object.
- Performance is symmetric, but not parallel. In this case the two hands are engaged in a task where both hands are doing work of similar temporal and spatial detail, however the two hands are not working at the same time - they are working serially. Real life examples include swinging on monkey bars and climbing a pole or rope.
- Performance is parallel, but not symmetric. In this case, both hands are working at the same time, however they are not doing work of the same level of temporal or spatial detail. Guiard's kinematic chain model describes some of these types of tasks and so handwriting is a real-life example (one hand is holding the paper while the other hand is writing). Other tasks that are asymmetric but do not follow Guiard's guidelines also fit in to this category (such as driving a standard transmission automobile).
- Performance is neither parallel nor symmetric. In this case the two hands are not working at the same time, nor are they working at similar levels of detail. This type of interaction boils down to single-handed interaction in which each hand performs different tasks in

serial. An example is playing chess. A player may remove a piece from the board using her left hand and then may move a different piece on the board to a new location using her right hand.

It is the first two cases that are relevant to this research. A successful symmetric interface will be one that affords parallelism as well as skill development and enhanced expressiveness. The amount of parallelism can be measured by timing the movement data of the two cursors under user control.

A number of different studies have shown the benefits of two-handed interaction for a variety of tasks. Buxton and Myers examined this issue in 1986 and demonstrated that in both a selection/positioning task and a text navigation/selection task, bimanual techniques outperformed unimanual techniques. In addition, the best performance times were linked to the subjects exhibiting the most parallel interaction [BM86].

2.2.2 Guiard's Kinematic Chain Model

By far the most referenced paper in the literature on bimanual interaction is Guiard's *Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model* [Gui87]. In this paper, Guiard describes higher order principles that govern asymmetric human action. These three principles are as follows (for a person who is right-hand dominant):

Right-to-Left Spatial Reference The left hand creates a frame of reference in which the right hand can act.

Left-Right Contrast in the Spatial-Temporal Scale of Motion The action of the right hand is faster and finer in spatial detail than the action of the left hand.

Left-Hand Precedence in Action The left hand's contribution to an action begins earlier than the contribution of the right hand.

The standard example given to demonstrate these principles is handwriting. The left hand begins the action and sets the frame of reference for the right hand by positioning the paper. The right hand then does the detailed, fast work of writing while the left hand occasionally adjusts the position of the paper. Obviously, there are many other activities that can be described by these principles. The question of importance to this research is whether or not these principles govern symmetric division of labour between the two hands. It seems obvious from my definition of symmetry that the second principle, that of contrast in the spatial and temporal scale of motion, will not apply to symmetric behaviour. It also seems unlikely that the first principle, that of right-to-left spatial reference, will apply. It seems more likely that the object being manipulated by the two hands will define the frame of reference for both

hands. Finally, it is uncertain that there will be any predetermined precedence of action in a symmetric interaction: the two hands may begin moving simultaneously. It seems plausible that if the two hands don't begin simultaneously, the precedence of action may be opposite to Guiard's principle. In other words, the dominant hand might begin the action, since the action is skilled, and the dominant hand is more accustomed to skilled activities.

In 1993 Kabbash et al. showed that the non-dominant hand is a poor approximation of the dominant hand for acquiring small targets [KMB93]. In 1994, Kabbash et al. studied four techniques for a compound drawing and colour selection task and found that an asymmetric Toolglass technique outperformed the other techniques [KBS94]. The other techniques included a second asymmetric technique, a unimanual technique and a bimanual technique in which the two hands performed independent tasks. This work showed the importance of design in bimanual interaction, because two-handed techniques can result in worse performance than single-hand techniques. None of the techniques tested in this work were symmetric. These results, combined with Guiard's model for asymmetric interaction has led HCI researchers to design and study asymmetric interaction techniques much more often than symmetric interaction techniques.

A variety of bimanual 3-D interactive systems for drawing, visualization, virtual reality and tele-medicine exist [SG97, HPPK98, ES01, HPP$^+$97, SG94, ZFS97]. These systems all follow the guidelines set out by Guiard for asymmetric interaction.

A number of specialized devices for bimanual input have also been developed. In particular, Balakrishnan and Patel investigated the use of a 'PadMouse' for the non-dominant hand [BP98]. The PadMouse consisted of a regular mouse with a two degree-of-freedom touchpad mounted on top of it. In the study described, the PadMouse was used by the non-dominant hand with the fingers accessing marking menus on the touchpad. The experimental analysis in this paper demonstrated the ability of the user's non-dominant hand to successfully use marking menus on the touchpad.

Fitzmaurice et al. created the 'Bricks' prototype of graspable user interfaces [FIB95]. Bricks contain 6 DOF input devices embedded in 1" cubes. Two of these bricks were used in a prototype two-handed drawing system and then embedded into the Alias Studio software. While the devices were identical and could be used together symmetrically, the operations employed in both the prototype and the commercial software were asymmetric. The two bricks played the distinct roles of 'anchor' and 'actuator'.

Raisamo and Räihä proposed a two-handed interaction technique for aligning objects in drawing programs using a straight-edge widget to push objects into alignment [RR96]. This method involved the straight-edge being translated by the mouse while the length of the straightedge (either vertical or horizontal) would be controlled by a trackball. Pressing the middle button on the mouse toggled the alignment of the straightedge between horizontal and vertical. This technique was clearly asymmetric as the right hand controlled the position of

the straightedge using the mouse. This research was followed by a generalization of the 'stick' metaphor for drawing, where the stick was used to rotate, cut and carve objects [Rai99]. However, these techniques were also performed asymmetrically, with the non-dominant hand using a large trackball to change the length of the various stick tools.

Early work on two-handed interaction by Chatty [Cha94b] led to the recommendation that any interface providing two-handed techniques be designed so that the task can be performed with only one hand, in case the second hand or device become unavailable. I do not agree with this recommendation. There are numerous activities that require both hands: lifting heavy items, measuring with a measuring tape and tying shoelaces, just to name a few. Chatty's recommendation was also made based on the premise that devices easily fail. The symmetric interaction that is described in this thesis are typical of those activities where two hands play equal roles. Also, the devices required (two USB mice) are stable, inexpensive and easy to replace, if necessary. Thus, Chatty's recommendation is not applicable to symmetric tasks that *require* the use of both hands.

Odell et al. compared toolglasses, marking menus and hotkeys as techniques for command selection for a shape drawing task [ODSW04]. Of the six techniques they tested, two were single-mouse techniques, two involved the use of a mouse combined with a keyboard and two were dual-mouse techniques. The two dual-mouse techniques they tested, bimanual marking menus and a toolglass implementation based on work by Bier et al. [BSP+93], gave the best and worst trial completion times, respectively. The surprisingly poor results of the toolglass technique led these authors to admit that the fine details are very important in the implementation of dual-mouse techniques. The bimanual marking menus, which gave the fastest results, were asymmetric in that the two hands were performing different tasks and at different levels of detail, but did not follow the asymmetric interaction guidelines set out by Guiard. The non-dominant hand was performing a separate task that did not set the frame of reference in which the dominant hand worked. The dominant hand began the interaction and the non-dominant hand followed. The interaction was asymmetric in that the dominant hand was doing work at a finer level of spatial detail. In general the results in this paper were controversial and quite questionable.

2.2.3 Symmetric Interaction

Although much work has been done on models of asymmetric bimanual interaction, such as Guiard's model, there has been much less research on models of symmetric interaction. There is no cohesive model that describes the way symmetric interaction works in the way that Guiard's model describes asymmetric interaction. A number of issues arise:

1. What types of tasks are suited to a symmetric division of labour?

2. When the division of labour allows symmetric interaction, what causes the action to degrade into asymmetric performance?

3. When performance is symmetric, what causes the action to be performed in parallel?

In *Symmetric Bimanual Interaction* [BH00], Balakrishnan and Hinckley begin to look at some of these issues. They present some experimental work in which subjects were asked to track two targets, each with one hand/cursor. They did not detect any statistically significant difference in tracking error between the two hands. In terms of the three questions posed above, Balakrishnan and Hinckley give some preliminary answers:

1. They state that any task where the two cursors could become widely separated would be unsuitable for symmetric interaction.

2. The only factor found to cause symmetric interaction to degrade into asymmetric interaction is a lack of visual integration in the task. Thus, in a two-handed rectangle drawing task, displaying just the rectangle corners instead of the whole rectangle leads to asymmetric performance.

3. They found that any one of the following three factors led to a degradation from parallel into sequential performance:

   - Lack of visual integration
   - Higher speed of task
   - Division of attention

Finally, Balakrishnan and Hinckley's findings indicate no tendency for the human motor system to devote more resources to the dominant hand when attention is divided while performing symmetric tasks. This is obviously quite different from the asymmetric bimanual interactions described by Guiard, and further indicates that a different model is needed to describe symmetric interaction.

Kurtenbach's T3 system [KFBB97] included many techniques that followed Guiard's guidelines. The T3 system also included a symmetric shape-creation technique called 'two-handed stretchies'. However, there was very little description of this interaction technique, especially as it pertained to editing pre-existing shapes, and no evaluation was reported.

Casalta, Guiard and Beaudouin-Lafon [CGBL99] performed a small study on two-handed rectangle drawing in which subjects were given both symmetric and asymmetric interfaces. In their studies, the subjects exhibited higher degrees of parallel activity with the symmetric interface than with the asymmetric interface. This shows that the right-to-left principle of the kinematic chain model does not hold for this activity. They state:

> Questions arise about the validity of the kinematic chain model in the special case of two-handed manipulation of a single object, as distinct from the more frequent case in which the two hands act on separate objects (e.g. the page in desk space and the pen in page space, in handwriting). [CGBL99]

Leganchuk et al. evaluated the two-handed stretchies technique of the T3 system for area-sweeping tasks [LZB98]. The evaluation compared two bimanual, symmetric area-sweeping techniques against a single-mouse area-sweeping technique. The bimanual approaches differed in that one used a toolglass technique for choosing the shape to use in the area selection task, while the other technique required the shape to be selected from a standard tearoff menu. There was no significant difference between the two bimanual techniques. The bimanual techniques were significantly faster than the single-mouse technique. In addition, the authors claim that the bimanual approach reduced cognitive load associated with the area-sweeping task because it was faster than the single-device approach even after the mode-switching times were removed from the single-mouse trial times. The devices used in this experiment were asymmetric: a stylus in the right hand and a puck in the left hand, both used on a Wacom tablet.

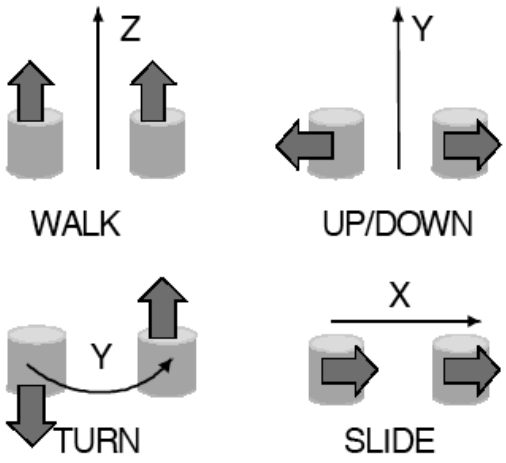


Figure 2.1: The bulldozer metaphor for navigating in a 3-D space. Image included with permission of Zhai.

In 1999, Zhai et al explored a 'bulldozer' style of interface that was extended to allow movement in a third dimension [ZKSS99] . The bulldozer technique used dual TrackPoint sticks (small joysticks that are typically found embedded in laptop keyboards). This interface was symmetric, and allowed users to navigate in a 3-D space 25-50% faster than with traditional single-mouse interaction techniques. The setup for this technique was specialized, using dual TrackPoints mounted on a wristpad. Figure 2.1 shows the extended flyable bulldozer metaphor.

One notable example of research that takes advantage of symmetric interaction is the symmetric deformation technique found in the Twister and Bender implementations for 3-D object manipulation by Llamas et al. [LKG+03, LPRS05]. They describe the ability to stretch and deform an object using symmetric interaction with two trackers, as well as some asymmetric interaction techniques. However, the emphasis in those research projects was on the mathematical models used for deformation of 3-D objects, rather than on the symmetry of the interaction. Their evaluation was based on the frame-rate achieved for real-time smooth deformations, and user performance and satisfaction were not measured.

Owen et al. [OKF+05] studied a symmetric interaction technique for manipulating spline curves. The purpose of their work was twofold: 1) to investigate whether bimanual input was affected by the two hands working in integrated versus separate workspaces and 2) whether the bimanual technique for manipulating spline control points was superior to the standard single device technique. Their results found that there was no difference between the workspace setup, i.e. the bimanual technique with the integrated workspace was not statistically different from the bimanual technique with separate workspaces.

Owen et al. did find that the bimanual techniques were faster than the single mouse technique. However, after the mode-switching time was removed from the single mouse trials, the single-mouse technique was faster than the bimanual techniques. This was counter to their hypothesis: they had hypothesized that the bimanual techniques would have both time-motion savings and cognitive benefit savings. While the time-motion savings were found, there were no cognitive savings, in fact, the single mouse technique allowed the users to perform the task faster, if the time required to switch between control points was ignored.

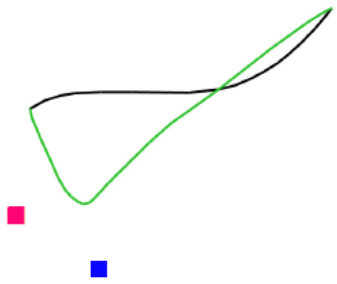


Figure 2.2: The spline manipulation technique in the Owen et al. study involved indirect spline manipulation where the participants manipulated control points, rather than directly manipulating the curve. Image included with permission of Owen.

An interesting aspect of Owen et al.'s research is that their bimanual approach used different devices for the two hands (a stylus in the right hand and a small puck in the left hand, both

used on a Wacom digitizing tablet). Thus, although the interaction was symmetric in that both hands were performing the same task (manipulating the 2-D position of control points), the devices were different. Using the same devices with the two hands might have led to better results for the bimanual techniques. Additionally, all three techniques tested were indirect. The task involved the manipulation of spline control points that were not located on the curve being manipulated, see Figure 2.2. Thus, there was no direct visual integration between the points being controlled in the bimanual configurations. Balakrishnan and Hinckley's [BH00] work on symmetric interaction show that a lack of visual integration will degrade symmetric interaction into asymmetric interaction. Thus, it seems that Owen et al.'s approach to bimanual spline manipulation was not well-designed.

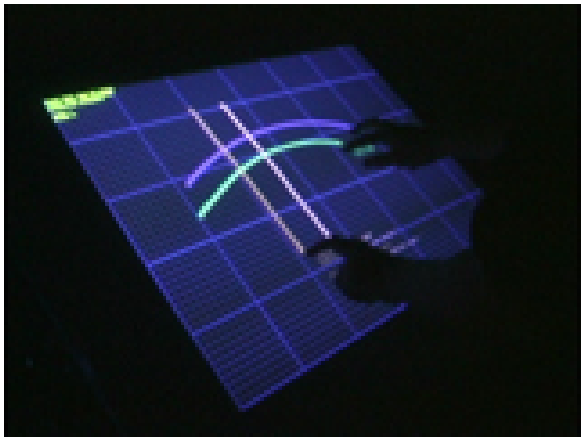


Figure 2.3: Multi-point interaction based on Frustrated Total Internal Reflectance.

Multi-touch interaction research has been ongoing for some time. At the SONY CSL lab, the HoloWall was developed to allow any part of the body to be used an an input device against a large input/output wall display [MR97]. Later, the same research group presented SmartSkin, an infrastructure for freehand manipulation on interactive surfaces, that allowed both hands to interact with digital objects [Rek02]. Commercial products are also available that allow designers to custom engineer multi-touch interaction: an example is Lemur, the multi-touch control surface from JazzMutant.[1] Recently, Han used frustrated total internal reflection (FTIR) to support multi-touch interaction [Han05]. This new technology for multi-touch interaction not only senses input from both hands, but senses input from each individual finger (Figure 2.3).[2] This technology merges the input and the output: the display surface *is* the input device. While the FTIR technology is still in its infancy, it has attracted attention from Apple. If this technology is incorporated into commercial products in the future, there

[1] `http://www.jazzmutant.com`

[2] Image source: `http://mrl.nyu.edu/~jhan/ftirtouch/`

will be even more need for a model of symmetric interaction.

2.2.4 Integrated Interaction

There are a few prototype research applications that have been developed that combine single device, asymmetric and symmetric interaction techniques. The previously discussed T3 system by Kurtenbach et al. [KFBB97] and a map navigation tool developed by Hinckley et al. [HCS98] are both examples of systems with multiple interaction styles.

Hinckley's map navigation task involved the use of a Wacom tablet with a puck in the non-dominant hand and a stylus in the dominant hand [HCS98]. Users could pan across a map by clicking and dragging the puck. They could annotate the map using the stylus. If they clicked with both the pen and the puck, they could symmetrically zoom in or out by dragging the devices together or apart, respectively. The authors called this a 'stretch and squeeze' technique. In addition, they could move the devices while stretching and squeezing to perform a compound zoom and pan. This system thus allowed single-device interaction with either the puck or the stylus or symmetric bimanual interaction using both. The compound zoom and pan technique was informally tested on six users and was reported as 'natural' and 'preferred' to a Microsoft mapping software package that used traditional mouse input.

In the same paper, the authors reported on a 'touchmouse' and touchpad technique for compound zooming and panning, but found this less successful than the stylus and puck technique. The touchmouse is a mouse that is adapted to sense when the fingers and hand are on the mouse. One of the difficulties with this technique was that the two different devices, the mouse and the touchpad, had different control-display ratios and so equal sized movements of the two hands did not result in equidistant motion on screen.

2.2.5 Summary of Previous Work in HCI

In summary, the previous work on bimanual input done by the HCI community has shown the performance benefits of bimanual interaction. This is especially true for asymmetric interaction techniques, which have been developed and evaluated for a variety of tasks and domains. There has been less work in the area of symmetric interaction, but the few studies that exist also show performance benefits over single-handed techniques. The Owen et al. study [OKF+05] found time-based performance benefits for the symmetric spline manipulation technique, but found that the cognitive load was lower for the single-mouse spline manipulation technique. However, there implementation did not provide visual integration between the two cursors.

Symmetric interaction needs to be evaluated with a wider variety of techniques and tasks, including tasks that go beyond the geometric manipulations that have thus far been implemented. In addition, it is necessary to provide a model of symmetric interaction techniques

Figure 2.4: The Robotron 2084 arcade game used two joysticks, one to control movement and the other to aim the gunfire.

and guidelines for implementation so that poor implementations of symmetric techniques can be avoided.

2.3 Bimanual Interaction in Gaming

While research in bimanual interaction has been ongoing in the HCI community, real bimanual interaction has been successfully commercialized in the gaming industry. Bimanual interaction has been part of physical games such as Labyrinth and Etch-A-Sketch, which both require symmetric use of the two hands. Historically, electronic games have required the use of a single hand and input device, although support for multiple players has meant that multiple input streams have been supported. Marble Madness is an early Atari game that used a single input device, consisting of a large track ball (in the arcade version). To get really fast motion in this game, players would roll the trackball quickly using both hands in quick succession. This is an interesting example of symmetric two-handed interaction using a single device.

Electronic game consoles have supported bimanual interaction for the individual player since the mid-1980s. Robotron 2084 used two joysticks; one joystick controlled movement and the other was used for aiming gunfire, see Figure 2.4.[3] Today's typical first-person-shooter games use two small thumb joysticks, one controls the movement of the player while the other

[3]Image source: `http://www.klov.com/`

Figure 2.5: Katamari Damarcy, a console game that uses a symmetric, dual-joystick tank-style interface to control the movement of the ball.

controls the viewpoint. This is clearly asymmetric in that the movement (sideways, forwards or backwards) is within the frame of reference of the viewpoint. There are also tank interfaces for some video games that are clearly symmetric. In these interfaces, both thumb joysticks control movement. Pushing forward with both thumbs is a forward motion, pushing backwards with both thumbs is a backwards motion. Pushing backwards with the left and forward with the right turns to the left, pushing forward with the left and backwards with the right turns to the right. The most popular example of this interface is Katamari Damarcy, in which players control a ball that picks up objects it runs over, and the object of the game is to pick up as much stuff as possible and have the ball get as big as possible, see Figure 2.5.[4]

These examples (and the success of the electronic console games) show that bimanual interaction is easily adapted by users, and users quickly become highly skilled in these games, whether the interaction is symmetric or asymmetric.

2.4 Bimanual Interaction in Psychology

Early psychology results in the area of two-handed cooperation [KSG79, Gui87] led many interface researchers to believe that the two hands cannot work together at a similar level of

[4]Image source: `http://www.crunkgames.com`

spatial or temporal detail because of interlimb motor interference. However, recent findings have shown that the two hands can in fact work together symmetrically [MKKP01, FZSW01, DIH+03]. These results show that problems of interlimb interference are associated with the planning and perceptual organization of the task being performed and have nothing to do with motor capabilities.

The amount of interlimb interference in two-handed dual target selection was first measured by Kelso et al. in 1979 [KSG79]. The authors measured the average total response time of 12 subjects reaching for a target with their left hand, reaching for a target with their right hand, and reaching for two targets using both hands. They found that when the two targets were located equidistant from the starting points, that dual hand target acquisition resulted in mean total response times appoximately 5% slower than the maximum of the response time with their left or right hand. The authors found more interlimb interference when the distances to the two targets varied in the dual target selection condition.

In 1996 Franz et al. found that callosotomy patients[5] don't have trouble drawing a line with one hand while drawing a circle with the other hand, but normal people do. This indicates that spatial interference is related to callosal connections.

In 2001, Franz et al. [FZSW01] showed that when a subject can perceptually integrate two unimanual tasks into a single, unified task, inter-limb interference is reduced and performance is improved. Additionally, Mechsner et al. [MKKP01] did a study of mirrored symmetrical and asymmetrical movements of homologous[6] and non-homologous muscle groups, using finger rotations. They found that subjects' movements converged to symmetrical finger rotations, whether the finger muscles being used were homologous or non-homologous. They assert that the inter-limb symmetry bias is perceptual, rather than related to muscle group activation, that perceptual goals are important, and that motor activity can be "flexibly tuned in".

Finally, in a recent journal article, Diedrichsen et al. [DIH+03] presented a number of experiments aimed at locating the source of bimanual interference in target selection tasks. They replicated earlier experiments on target selection and found that interlimb interference comes from the selection or decision process rather than during motor programming. When the targets to be selected were directly cued, rather than abstractly encoded, the inter-limb interference virtually disappeared.

These results indicate that, in certain circumstances, users are capable of using their two hands together at similar levels of spatial and temporal detail, contrary to the generally-held belief of the user interface community. This suggests that symmetric bimanual interaction in the computer interface has great potential, provided the tasks perceptually unify the two hands.

[5] A callosotomy patient is someone who has had the connections between the two halves of the brain surgically split.

[6] Homologous muscle groups are the same muscle groups on each side of the body, such as the index finger on both hands.

2.5 Bimanual Interaction in Art and Education

Johannes Itten was a teacher at the Bauhaus from 1919 to 1922, during which time he developed the innovative Preliminary Course, which was designed to teach students the basics of design and color [Wic00]. Itten's approach to teaching was highly individualistic, even spiritual. Itten's pedagogy departed radically from academic training in its emphasis on the harmonization of mind and body. His training included two-handed, symmetrical drawing exercises to help students to train the coordination of their motor skills reduce the deficit of dexterity in the non-dominant hand and promote intuitive response. Itten's pedagogy is related to the pedagogical reform movement of Maria Montessori and others.

The Montessori education method uses the Brain Gym *double doodle* and the Brain Gym *lateral eight* exercises to promote lateral movement across the midline of the body [DD94]. The double doodle exercise consists of having students doodle at a large board using both hands. The student is instructed to draw the same thing with both hands, such that the drawings are a reflection across the vertical midline of the body. The lateral eight exercise involves students drawing a large infinity symbol (large enough that they fully stretch their arms while drawing), first with one hand, then with the other, and then with both hands together. These exercises show the importance of symmetric interaction in other fields.

2.6 Previous Work in Dual-Mouse Systems

The symPut driver that will be introduced in Chapter 3 is not the first system to allow multiple spatial input streams. Other systems that afford multiple input streams are described here. These libraries were not intended to support a single user operating two mice simultaneously. Generally, these libraries have been created to support development of computer supported collaborative work (CSCW).

2.6.1 Multiple Input Device (MID) Java Toolkit

Hourcade and Bederson created the Multiple Input Device or MID toolkit [HB99] to afford multiple mouse interaction for collaborative or group interfaces. The MID toolkit was built in Java and could run only on Windows 98/NT. When Microsoft released Windows 2000, the MID toolkit became unworkable. This toolkit was used to create collaborative applications such as Kidpad, a children's story-telling application [BBA+00].

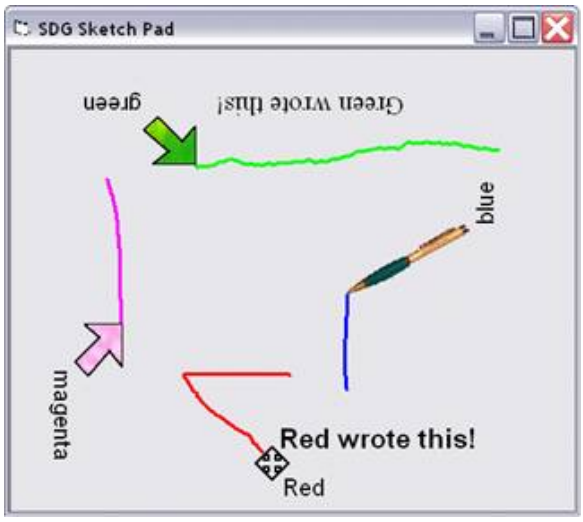


Figure 2.6: SDG Sketch Pad: the collaborative table-top drawing program created using the Single Display Groupware toolkit.

2.6.2 SDG Toolkit

Tse created the SDG (Single Display Groupware) Toolkit [Tse04, TG04] based on Bederson's MID toolkit. The SDG toolkit is in use today and works on Windows platforms. Using the SDG toolkit requires programming the application in the .NET environment. As with MID, the SDG toolkit is designed for supporting multiple mice and keyboards for collaborative work, rather than for a single person. Thus, the framework is not based on ideas of symmetric or asymmetric interaction but rather group collaboration. The SDG toolkit works by using the RAW input layer accessible under Windows XP. A screenshot of a collaborative, table-based sketch application built on top of the SDG toolkit is shown in Figure 2.6 [7].

2.6.3 ManyMouse Library

The 'ManyMouse' library [Gor] is a free, multi-platform library for supporting multiple mice and is designed for gaming and other collaborative applications. The library works on Macintosh, Windows and Linux operating systems. It does not easily support symmetric input, but can be configured to do so. The biggest drawback to using the ManyMouse library for single user, symmetric interaction applications is that the library does not 'seize' control of the mice. So, all mice control the system cursor while providing an accessible input stream to

[7] Image source: `http://grouplab.cpsc.ucalgary.ca/software/SDGT/index.html`

the including application. Thus, as the user interacts with the application, the system mouse is also generating spurious events.

2.7 Previous Work Summary

In summary, symmetric interaction techniques have not been studied as extensively as asymmetric techniques, but the few studies that have examined symmetric interaction have reported positive results. While early psychology findings led to the belief that there would be motor interference if the two hands were assigned tasks of equal difficulty, more recent work have shown that if the two hands are cooperating on a perceptually unified task, motor interference will not be an issue. These factors point to the potential performance gains that symmetric interaction techniques might offer. While some other toolkits have been implemented to support multiple pointing devices, none have been developed to specifically enable symmetric interaction techniques, while still allowing for asymmetric interaction.

Chapter 3

The Symmetric Interaction Model

Guiard's Kinematic Chain model was described in Chapter 2 and is well established. However, it provides no guidance for interface designers when faced with a task that is naturally symmetric. Thus, there is a need for a model of symmetric interaction which both helps to identify tasks that can benefit from symmetric interaction and provides guidelines for implementing symmetric interaction techniques. The Symmetric Interaction Model presented in this chapter addresses these needs. Chapters 4, 5 and 6 provide descriptions of applications and techniques that have been built based on the Symmetric Interaction Model. Those chapters also contain evaluations of the presented techniques that show the performance benefits of using symmetric interaction techniques for symmetric tasks.

The Symmetric Interaction Model is based on my philosophy of how two-handed symmetric interaction works. This philosophy is a synthesis of knowledge gained over five years of implementing and testing symmetric interaction techniques and applications. The rest of this thesis is meant to support this view of symmetric interaction. The model comprises five parts that are presented in this chapter:

1. A taxonomy of symmetries in the user interface.
2. Symmetric task classifications.
3. Rules of thumb for the symmetric versus asymmetric design decision.
4. The symmetric-asymmetric connection.
5. Guidelines for implementing symmetric interaction techniques.

Most users of laptop computers are aware of what happens when a second spatial input device is plugged into a computer, because they have used an external mouse with their laptop. The mouse and the built-in spatial input device, whether it is a touch-pad, a track-ball or a touch-point, fight for control of the cursor on screen. The operating system combines the spatial input from the two devices into a single input stream that defines the interaction between the user and the system. Although the user has two hands and could benefit from using two input devices to control two cursors, the operating system prevents it. This is true of Microsoft Windows, Macintosh OS and the various flavors of UNIX and Linux. The symPut driver, described in Section 4.2, addresses this issue and allows two spatial input streams to be generated from two separate mice.

3.1 Taxonomy of Symmetries

The term symmetry is overloaded. Webster's Ninth New Collegiate Dictionary [MW86] provides four definitions of symmetry:

1. Balanced proportions; also: beauty of form arising from balanced proportions.
2. The property of being symmetrical; esp: correspondence in size, shape, and relative position of parts on opposite sides of a dividing line or median plane or about a center or axis.
3. A rigid motion of a geometric figure that determines a one-to-one mapping onto itself.
4. The property of remaining invariant under certain changes (as of orientation in space, of the sign of the electric charge, of parity, or of the direction of time flow – used of physical phenomena and of equations describing them.

The first two definitions of symmetry relate to a symmetry of proportion or balance between parts of a whole and a geometric, reflective symmetry. These two types of symmetry are both relevant to this work. They are used in a variety of ways and can be found in many forms in the applications described in this thesis. In this section, I define and discuss the various forms of symmetry that will be relevant to the reader.

User Symmetry: User symmetry refers to the reflection of affordances across the vertical mid-line of the body. This symmetry is only approximate, because both genetic and learned factors break this symmetry. Hand dominance is an example of a genetic factor; writing, which lacks reflection symmetry, is an example of a learned one. Despite the fact that this symmetry is approximate, it exhibits strong influences on our interactions and especially in the way our two hands cooperate, as evidenced in the work by Mechsner et al. [MKKP01].

Device Symmetry: Device symmetry refers to the use of identical devices by both hands. Users of a system can only exploit symmetries provided by the input devices of that system. The computer keyboard affords symmetric interaction: both hands are proportionately involved in the activity. As Mackenzie and Guiard point out, this symmetry is not perfect, due to the imbalance in the placement of the power keys on the keyboard [MG01]. However, it is balanced across the two hands for the entry of alpha-numeric keys. In standard desktop computer systems, the use of a single mouse does not provide device symmetry. A single mouse is located on one side of the keyboard, making assymmetric the affordances available to the two hands of the user. While the mouse has no special position, and can be used on either side of the keyboard, the mouse buttons themselves have meaning in most applications. A user who wants to move the mouse to the left of the keyboard to use with his left hand will have to adjust software settings in order for the buttons to be mapped so that the main button is under his index finger.

Actuation Symmetry: Actuation symmetry refers to the ability of a user to initiate or execute program functionality using either hand. A user interface offers mechanisms for activating affordances, which are triggered by users manipulating devices. For a mouse-activated affordance, actuation symmetry is possible only if two mice are present, thus actuation symmetry *requires* device symmetry. It is interesting to note that the standard computer keyboard does not provide actuation symmetry. The key used to execute most commands, 'Enter', is located only on the right side of the keyboard.

Bilateral Function Symmetry: Bilateral function symmetry refers to the assignment of actuator (button or key) functionality that is symmetrically reflected across the vertical mid-line of the body. The psychology literature on stimulus-response compatibility shows that there are certain mappings between left and right hand control and two-handed tasks that are easy. These tasks become much harder when the mappings are switched [MKKP01, FD54, NU84, CTS+97]. Mappings that are symmetric across the vertical mid-line of the body tend to fall into the easy category, asymmetric mappings tend to be more difficult. Bilateral function symmetry takes these findings into account and requires homologous muscle group usage when actuating the same function with the left and right hands. Thus, if users activate function A by pressing the index finger on their right hand, they must also be able to activate function A by pressing the index finger on their left hand (rather than pressing the ring finger). This mapping is intuitive for users — they know that pressing the button under either index finger accomplishes the same task. This type of mirrored mapping was shown to be very effective for the Half-QWERTY keyboard [MMB93] because it is the finger muscles used to hit the key that the brain remembers, rather than the spatial location of the key. This also follows the finger numbering system used in teaching piano, as shown in Figure 3.1.[1]

[1] Image source: `www.umkc.edu/is/mts/kcen/piano/`

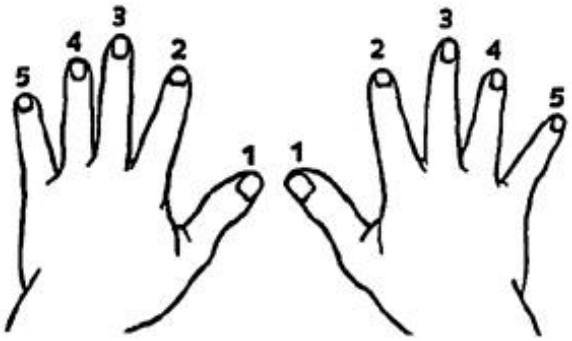


Figure 3.1: Finger numbering for learning to play piano.

Interaction Symmetry: Interaction symmetry refers to a proportional balance between the amount of work performed by the two hands and the level of difficulty of work performed by the two hands. A user interface that provides interaction symmetry is one that allows the users' two hands to work together, at similar levels of spatial and temporal detail, on objects in the program. Interaction symmetry is thus a statistical symmetry, an accounting of the similar levels of detail of the affordances granted to the two hands. Thus, a *symmetric interaction technique* is defined as a technique that requires interaction symmetry. Interaction symmetry is the most important type of symmetry related to the work in this thesis.

Reflection Symmetry: Reflection symmetry refers to identical actions or motions of the hands and/or arms reflected across the vertical mid-line of the body. The way that children clap their hands, with the hands vertical and centered in front of their bodies, is a typical example of reflection symmetry. While reflection symmetry may be present in the use of symmetric interaction techniques, it is not required for the technique to be considered symmetric.

Presentation Symmetry: Presentation symmetry refers to a symmetry in the affordances presented to the user on a display, so that there is no left or right bias in the placement of widgets, tools or information displays. The display of a user interface consists of two main parts: the widgets that allow the user to act on the data, and the data itself. The layout of the widgets can be left-right reflective, so that any tool available on the left side of the screen is also available on the right side of the screen. A presentation that is not left-right reflective is one that has built-in left or right hand biases. Presentation symmetry does not require device symmetry or actuation symmetry, but works very well in conjunction with both of these symmetries. It is important to note that having the same widget available on both sides of an interface makes more sense when there are two trackers controlled by the two hands so that a user can interact with a widget with

whichever cursor is closest to that widget. Of course, having multiple copies of the same widget on screen is not likely to be desirable due to screen real estate scarcity. The use of popup menus for accessing program functionality is an option that maintains presentation symmetry and maximizes screen real estate. In general, having fewer widgets on screen will both increase the amount of screen real estate devoted to the user's data and enforce a simpler and more elegant interface design.

The taxonomy of symmetries is important to keep in mind when designing both symmetric interaction techniques and applications that include symmetric techniques. The types of symmetry described in the taxonomy must be supported by the computer system to allow symmetric techniques and applications to be built. These types of symmetry are not supported in standard operating systems (which don't allow separate spatial input streams). Thus, the symPut driver, which will be presented in Chapter 4, was built to support this taxonomy.

3.2 Symmetric Task Classifications

The Symmetric Interaction Model is a model to help interface designers develop symmetric interaction techniques. But symmetric techniques are only suitable for tasks that are naturally symmetric. Thus, it is necessary to be able to determine what a symmetric tasks look like. One way to do this is to look at common tasks from everyday life and determine whether or not they are symmetric. This is not as simple as it seems. Although some tasks (such as clapping and skipping) are clearly symmetric, and some tasks (such as handwriting) are clearly asymmetric, there are other tasks that do not clearly fit into one category or the other (such as playing golf or hockey[2]).

Some compound tasks have sub-tasks that are naturally symmetric and sub-tasks that are naturally asymmetric or even one-handed. An example is driving a car with a standard transmission. While on the highway, a driver may use both hands symmetrically to steer the car. When switching gears, the left hand is used to stabilize the steering wheel while the right-hand adjusts the gear shift. To complicate matters further, some tasks are more symmetric as the skill-level of the person performing the task increases. In this section, I present two classifications that show how some common tasks should be categorized with respect to the level of interaction symmetry and the skill level required to perform the task.

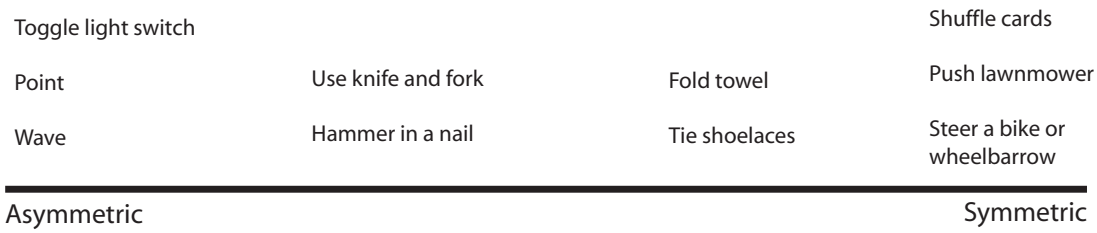


Figure 3.2: The Symmetry Spectrum, illustrating everyday tasks.

3.2.1 The Symmetry Spectrum

There are tasks that are clearly symmetric and tasks that are clearly asymmetric and some tasks for which the amount of symmetry is unclear. Thus, there is actually a spectrum of symmetry, and any task can be placed somewhere along the spectrum. The Symmetry Spectrum is illustrated in Figure 3.2. At the left end of the spectrum are tasks that are purely asymmetric, including one-handed tasks. At the right end of the spectrum are tasks that are purely symmetric. The tasks illustrated in the Symmetry Spectrum are everyday, household tasks that do not require high levels of skill.

Figure 3.3 takes the Symmetry Spectrum and applies it to the field of human-computer interaction and mouse-based input. The varying ways that mice can be used by themselves, with other devices and with other mice are shown relative to the level of symmetry they afford.

3.2.2 The Symmetry-Skill Plane

The Symmetry Spectrum for everyday tasks does not fully capture the differences in tasks. In the spectrum in Figure 3.2, playing a Bach invention on the piano would be placed at the same point on the spectrum as hand-clapping. Both require equal levels of detail in the temporal and spatial actions of the two hands. However, there are two major differences between these two tasks. The first, and most obvious, difference between hand-clapping and playing a Bach invention is the level of skill required — most one-year-old infants have mastered hand-clapping, only some very accomplished pianists master the Bach inventions and only after years of practice.

[2]In both golf and hockey, the player's hands are both involved in holding the club or stick, and both contribute to the use of that implement. In terms of timing and movement, the hands are contributing almost equally, but it seems likely that the dominant hand is leading. Thus, these tasks fit somewhere in the middle of the symmetry spectrum.

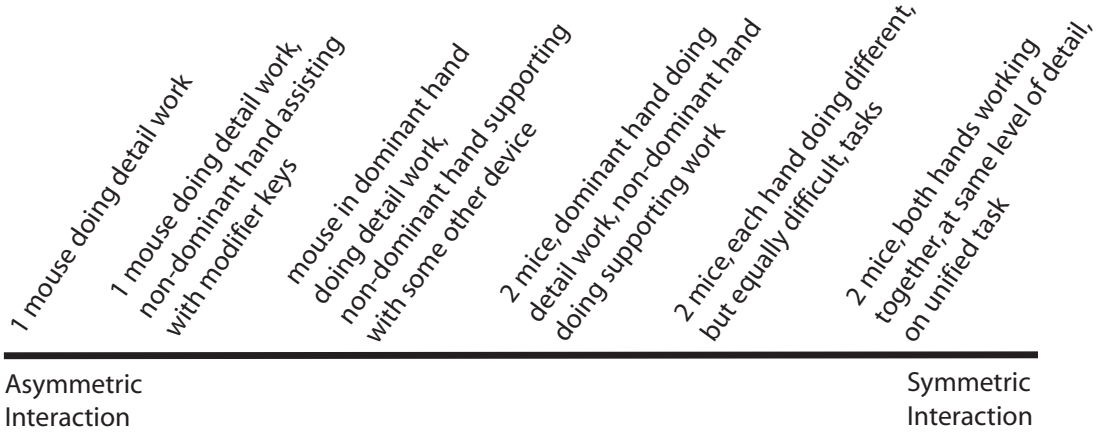


Figure 3.3: The Symmetry Spectrum for HCI, illustrating the devices and how they can be used in human-computer interaction.

The second difference between hand-clapping and playing a Bach Invention is the reflective symmetry in the action of the two hands. In hand-clapping, the hands are doing exactly the same motion, mirrored across the vertical mid-line of the body. In a Bach invention, the two hands are not performing symmetrically mirrored tasks. When the pianist presses a key with Finger 1 on the left hand, she may have to simultaneously press a key with Finger 4 on the right hand (see the piano finger numbering in Figure 3.1). Additionally, the two arms and hands may not be spaced equally away from the vertical mid-line of the body — the right arm and hand may be extended further up the piano keyboard with the left arm and hand almost in front of the body.

The differences in skill level and reflective symmetry between hand-clapping and playing a Bach invention demonstrate that the Symmetry Spectrum is not enough to fully classify tasks for the purpose of designing interaction techniques. To capture these more intricate, but important details, I have created a two-dimensional model of task space, called the Symmetry-Skill Plane, see Figure 3.4.

The Symmetry-Skill Plane has the interaction symmetry of the task represented along the x-axis. Recall that the interaction symmetry refers to similar levels of temporal and spatial action performed by the two hands. Tasks that are statistically asymmetric are located closer to the origin on the x-axis, while tasks that are statistically symmetric are located further away from the origin. The skill level required to perform the task is represented on the y-axis, with 'easy' tasks closer to the origin and tasks requiring intense practice and skill development further away.

While it is possible to build a three dimensional categorization of tasks, with the amount of

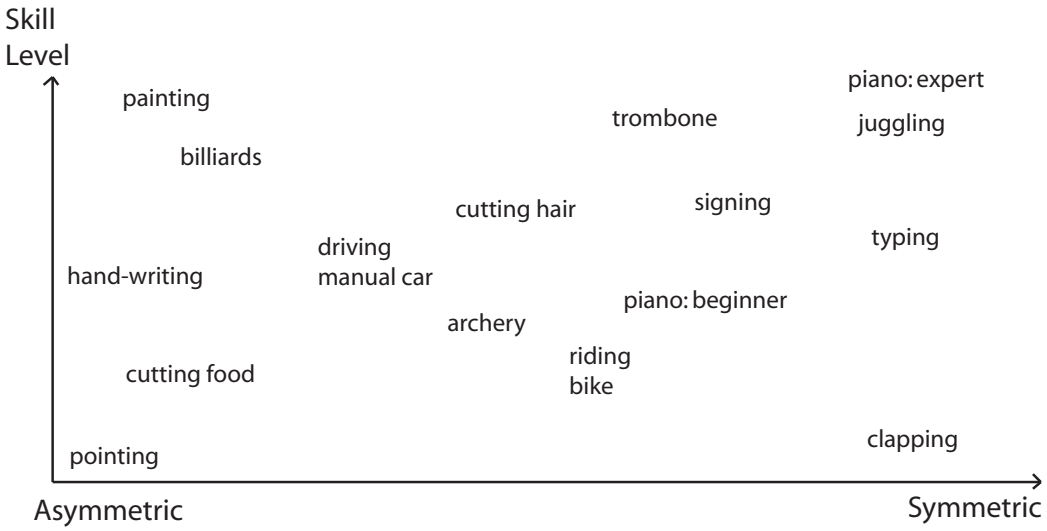


Figure 3.4: The Symmetry-Skill Plane: tasks can be placed horizontally according to the level of statistical symmetry and vertically according to the skill level required. Of course, some tasks can be performed at a variety of skill levels. Placement in the plane is only approximate.

reflective motion as the third axis, such a categorization does not enlighten the design decisions or a user interface developer. Whether or not a user makes mirrored movements during task performance should not affect the design of the technique for that task.

3.2.3 Domain Specific Symmetry-Skill Planes

In any given task domain, there can exist a large variety of tasks that may vary in both difficulty and symmetry. The domain of piano music is a good example. There are pieces of piano music that are clearly simple (for an individual who has had basic piano training) and can be played with one hand. There are also pieces of music (such as Bach Inventions) that require years of practice and skill development and are very much spatially and temporally symmetric. And, there are many pieces of music for the piano that fit somewhere between these two extremes. There are some pieces of music that involve a reflective symmetry in the actions of the two hands, but most do not. Figure 3.5 shows a symmetry-skill plane with some sample pieces of music to demonstrate this variety.

The variety of symmetry in music pieces can be echoed in the specific examples of many

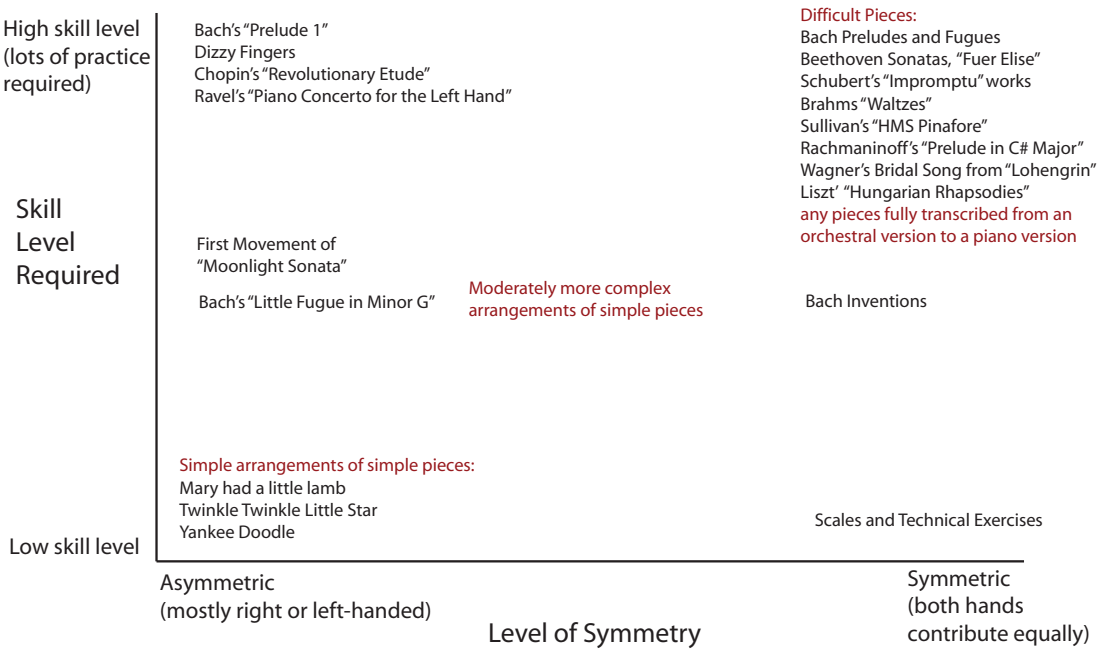


Figure 3.5: The Symmetry-Skill Plane for piano music.

tasks. A dancer is very likely to demonstrate a high degree of interaction symmetry in his movements during a given dance. However, depending on the specific dance, he may exhibit more of less reflective symmetry, and he may require more or less technical skill.

These examples confound the process of designing interaction techniques for symmetric tasks in the human-computer interface. Not only does a designer have to determine whether a task is symmetric or asymmetric, they have to determine whether it is a task that requires skill to perform. Many tasks will have a different level of interaction symmetry depending on the skill level of the user performing the task. Because of this, there should be a way to transition a user from unskilled, asymmetric performance of a task to skilled, symmetric performance. Another form in which the lack of skill may manifest itself is in the level of parallelism. A novice user of a symmetric technique may use both hands equally, but may use each hand serially, switching back and forth between the use of the two hands. As skill develops, the level of parallelism in task performance will increase.

3.3 Symmetry or Asymmetry?

An important issue to resolve if interaction designers are going to consider developing symmetric interaction techniques is how to determine whether a task should be performed symmetrically or asymmetrically. As described in Section 3.2.2, there are certain tasks that are clearly symmetric and clearly asymmetric. But many tasks fit somewhere in the middle of the Symmetry Spectrum. In these cases, it may be much more difficult for a designer to decide between a symmetric and an asymmetric implementation.

Additionally, there may be tasks which have been historically performed in the user interface with a single spatial input device. Because of the limitations of the single spatial input device, a task that is naturally symmetric is broken down into sub-tasks that are performed serially by one hand operating a single device. A prime example is the translation and rotation of objects. When a person wants to move a stack of papers from the center of their desk to the corner, and turn the stack of papers by ninety degrees, it is unlikely that the person will move the papers to the corner and then turn them. The person is much more likely to turn the papers *while* moving them. This task is naturally symmetric, and may be done with two hands if the stack of paper is heavy or with one hand (making use of wrist rotation) if the papers are light. In either case, the task is perceptually unified, and the user does not think of the translation and the rotation as separate tasks. In the user interface, this task is broken down into separate sub-tasks of rotation and translation, simply because of the limitations of a single spatial input device. Most computer users are completely accustomed to this task breakdown and do not give it a second thought. Thus, when methods of interaction involving unnatural sub-tasks are historically ingrained, it may be difficult to recognize the real task as symmetric.

Another difficulty faced by designers is in domains where there is no real-life task to evaluate. Data mining is an application that was not possible before the emergence of large databases and high-performance computing. It is possible that there are ways of interacting with very large datasets that are suitable for symmetric interaction. However, a user interface designer for a data-mining application will not have a real example to use as a guide in placing a data-mining task on the symmetry spectrum.

Thus, it is necessary to provide some rules of thumb for determining the suitability of a symmetric or asymmetric interaction technique for a given task. The rules given here can be thought of as characteristics of a naturally symmetric task. Not every symmetric task will exhibit all of the characteristics described below, but most symmetric tasks will exhibit some of them. If the task under consideration exhibits none or only a few of these characteristics, it is likely more suitable for an asymmetric (either single-device or dual-device) technique.

- The task involves exploration of a multi-dimensional domain space.

- Performing the task with a single spatial device would require constant mode-switching.
- The two hands can be used together to manipulate a single object.
- The task involves the geometric manipulation of objects (scale, translation, rotation, shear).
- The task is similar to tasks listed at the symmetric end of the Symmetry Spectrum or the Symmetry-Skill Plane.
- The task cannot be split into two sub-tasks that are identifiably 'easy' and 'hard'.

The first two items in this list are more applicable to deciding whether the task should be performed with two hands and two devices or with a single hand. The last four items are specifically related to the symmetric versus asymmetric decision. The final rule in the list is likely to be the most useful. If a designer attempts to split a task into two sub-tasks and one of those sub-tasks is not obviously harder than the other, it is likely that the task is naturally symmetric, and requires equal contributions from both hands. An alternative way of stating this rule is to say that if the sub-tasks are different, but it's unclear which task should be given to the dominant hand and which should be given to the non-dominant hand, then the overall task is probably symmetric. In this case, splitting the overall task into two equal sub-tasks, where the two hands are doing the same thing, would be a better solution. This is likely to lead to less cognitive load because the user will not have to remember which hand is performing which task.

3.4 The Symmetric Superset

A computer that is set up with two symmetric devices and runs an application allowing dual cursors is able to support symmetric, asymmetric and single-cursor interaction techniques. The application may or may not use all three styles of interaction, but the symmetric device setup affords this. The two devices can be used together to symmetrically control an object on screen. The devices can be used asymmetrically, with one device controlling a zoom feature while the other device does detailed editing or object manipulation. And, either device can be used on its own as a single device for object selection or sketching. Thus, a computer setup with symmetric spatial input devices and dual cursors in an application affords a 'superset' of interaction. The word superset refers to the generality of the interaction style: asymmetric interaction can be thought of as a special case of symmetric interaction. A general computing setup with symmetric devices can afford symmetric and asymmetric interaction.

A system setup with different spatial input devices, such as a pen and tablet and a trackball is unsuited to symmetric interaction. Asymmetric devices do not afford symmetric interaction

because they can not provide actuation symmetry. In addition, using different devices in the two hands can lead to difficulties because of differences in the control-display ratio, as demonstrated in the symmetric map navigation interaction technique presented by Hinckley et al. [HCS98] and described in Chapter 2.

A computer with a single spatial input device is not usually capable of providing symmetric interaction, although there can be exceptions. A specialized multi-touch screen affords symmetric interaction, and some input devices (such as large trackballs) can be used with both hands together. Although the keyboard can be used in conjunction with the single spatial input device to provide asymmetric interaction, this level of asymmetric interaction is limited. Thus, a computer set up with symmetric devices and capable of handling separate input streams from those devices to generate dual cursors is the most versatile setup because it affords the union of symmetric, asymmetric and single-cursor interaction techniques.

3.5 Guidelines for Implementing Symmetric Interaction Techniques

Once a designer has made the decision to implement a symmetric interaction technique for a given task, he or she must actually design the interaction. Here I present a set of guidelines on how to do that. Although some of these guidelines were originally derived from the objectives of the Symmetric Interaction Model, some of them were added during the iterative development of the symmetric techniques presented in later chapters.

It is important to note that these guidelines are only starting points, and they have not been proven empirically. They are based on my experiences designing and evaluating symmetric interaction techniques. These guidelines should be challenged and empirically tested in the future.

Obviously, it is important before designing a symmetric interaction technique to choose a task that is suitable for interaction symmetry. Tasks that involve the manipulation of an object, where the manipulation can be performed with two cursors, are ideal. Once a task is identified as naturally symmetric, the following guidelines should be followed in designing a symmetric interaction technique:

Actuation Symmetry: Make all program functionality available to both hands. This ensures that the interface is accessible to users from across the handedness spectrum, as users can actuate program functionality with whichever hand they prefer.

Bilateral Function Symmetry: Use a left-right reflective symmetry in the actuator functionality on devices. Buttons pressed by the same finger on either hand should have identical functionality.

Device Symmetry: Use identical devices for each hand, to afford actuation symmetry and bilateral function symmetry. Note from the discussion of the symmetric superset (Section 3.4) that device symmetry will also afford asymmetric interaction and single-device interaction that is free of handedness bias.

Frame of Reference: The two hands should work in the *same* frame of reference.

Popup menus: Use transparent menus that pop-up where needed; this will afford interaction symmetry and maintain the presentation symmetry of the program interface. The transparency will prevent the user's data from being obscured and help the user maintain task context. Actuation symmetry requires that menu activation be available via either hand.

Start/end Allowances: Allow users to start any action with either hand and to end any action with either hand. Users should also be able to start and end actions with both hands simultaneously.

Visual Integration: When the two hands are working symmetrically the cursors must be visually integrated by a connecting line or other means. Manipulation of a single object by the two hands is optimal, because no artificial connection need be introduced.

Presentation Symmetry: Remove widgets from the screen space wherever possible and make all program functionality available through the use of popup menus. If it is necessary to display widgets on screen, they should be centered at the top or bottom, or present at both sides of the screen. This symmetry means that either cursor can be easily used to access program functionality, which helps to make the interface free of handedness bias. In many cases, screen real estate is scarce and therefore popup menus are a better choice than replicated static widgets for making the interface free of handedness bias.

Cursor Design: Differentiate the cursors with shape as well as colour. Differentiating the cursors by colour alone is not sufficient to help users know which hand controls which cursor. Good choices for cursor shape include left and right hands or arrows that point toward the center of the screen.

Horizontal Movement Restriction: If it is suitable, prevent the cursors from crossing one another horizontally on screen. This will prevent stimulus-response incompatibility issues. However, this is not possible if object rotation is required.

A purely symmetric interaction technique will *require* the use of both hands. This has the benefit of helping the user to develop mousing skill in their non-dominant hand, without requiring the user to consciously make an effort to do so. Novice users may perform a symmetric technique serially at first, but will work more in parallel as they gain skill.

3.6 Summary

Symmetry is an overused word, and its application to bimanual input has given it a special meaning that is related to the statistical use of the two hands (both temporal and spatial) in an interaction over time. Thus, *symmetric interaction* refers to interaction in which the two hands work together at similar levels of temporal and spatial detail. However, because symmetry is an overloaded word, it is necessary to provide a taxonomy of symmetries that are relevant to the user interface. Having defined these symmetries, it is possible to build a model for determining how symmetric an interaction is. This can be determined in a simplistic way by placing a task along the provided Symmetry Spectrum. More insight into the task characteristics can be gained by placing the task in the Symmetry-Skill Plane. These classifications are provided to help designers determine the relative symmetry of a task and to then choose an appropriate interaction technique for performing that task in the user interface. The Symmetric Interaction Model can be summarized by the following four points:

1. Symmetric interaction is a superset that includes asymmetric interaction. An interface that affords symmetric interaction can be used asymmetrically, but the reverse is not usually true.

2. Symmetric interaction requires symmetric devices, or a specialized single device that allows both hands to interact together.

3. Symmetric interaction requires bilateral function symmetry. Pressing a button with a given finger on one hand should activate the same functionality if the same finger on the other hand presses a button.

4. Symmetric interaction should involve a unified task, where the two hands work together to manipulate a single object. This means the two hands should work in the *same* frame of reference.

In Chapter 4, I present a driver, called symPut, that I created to support the development of symmetric interaction techniques and applications. This is followed by a description of a 2-D symmetric drawing application, symDraw, that was designed based on the Symmetric Interaction Model. Chapter 4 also includes some experimental analysis of techniques for object manipulation, which provides evidence to support the hypothesis presented in Chapter 1. Chapter 5 will show how the Symmetric Interaction Model presented here was used to create a completely new technique for manipulating spline curves. Chapter 6 provides an example of how the Symmetric Interaction Model can be applied to develop symmetric bimanual techniques for non-spatial tasks.

Chapter 4

Basic Geometry Manipulation

The previous chapter presented the Symmetric Interaction Model. This chapter presents the symPut driver that I built to support the development of symmetric interaction techniques and applications. The symPut description is followed by a description of the first application developed using the symPut driver: a dual-cursor 2-D drawing program called symDraw. The symDraw application includes a number of symmetric interaction techniques for shape creation and manipulation. While these techniques are not new (they were first described by Kurtenbach et al. in the T3 system [KFBB97]), I developed independently for this application. Additionally, the symmetric object manipulation techniques were never evaluated by Kurtenbach et al. This chapter presents two formal experiments which test the techniques against both dual-mouse asymmetric techniques and single-mouse techniques.

4.1 Previous Work in Bimanual Geometry Manipulation

Kurtenbach et al. implemented symmetric shape creation techniques, called 'two-handed stretchies', in the T3 system [KFBB97]. These were implemented as part of a large 2-D graphics editing system that encompassed mostly asymmetric interaction techniques. Leganchuk et al. studied area-sweeping tasks, which can be considered functionally equivalent to the 'two-handed stretchies' version of rectangle creation. They found that the symmetric technique outperformed other techniques [LZB98] and that there was lower cognitive load associated with creating rectangles symmetrically than with one device. Carson et al. did a study in which subjects were asked to draw circles on paper simultaneously with each hand [CTS+97]. Movement distortions and phase wandering were problems when the subjects were asked to draw both circles clockwise or counter-clockwise, but not when the subjects were asked to draw the circles symmetrically (one hand drawing clockwise, the other hand drawing counter-clockwise).

As expected, the distortions measured in the asymmetric test were most evident in the non-dominant hand. These results suggest that a mirrored symmetric bimanual interface for drawing is likely to be more intuitive and allow the non-dominant hand to perform better than an asymmetric bimanual drawing interface.

In summary, although some work has been done to evaluate symmetric shape *creation* techniques, there has been no evaluation of symmetric shape *manipulation* techniques. The experiments presented in this chapter address that void.

4.2 The symPut Driver

The symPut driver was originally developed in C/C++ for a Linux system. It has since been ported to work through the X11 interface on the Macintosh OS X platform. The symPut driver was designed to optimize the ease of symmetric interaction in application development. As discussed in Section 3.4, by affording symmetric interaction, the driver also affords asymmetric and single-cursor interaction techniques.

The symPut driver makes use of the event module in Linux. When USB mice are attached to the system, the input streams from these mice are forwarded to separate device files in the /dev/input/ directory. These files can then be polled and read separately. The symPut driver reads these events and then sends the information to an application that has included the symPut library. Under the Macintosh system, a custom multiple input driver polls for events and sends them to the application.

An application makes use of symPut by including symPut.cc at compile time. In order to use symPut, the application must provide implementations of the procedures listed below.

- void mainButtonDownEvent(int dev)
- void mainButtonUpEvent(int dev)
- void middleButtonDownEvent(int dev)
- void middleButtonUpEvent(int dev)
- void outerButtonDownEvent(int dev)
- void outerButtonUpEvent(int dev)
- void relativeXmovement(int dev, int value)
- void relativeYmovement(int dev, int value)
- void wheelEvent(int dev, int value)

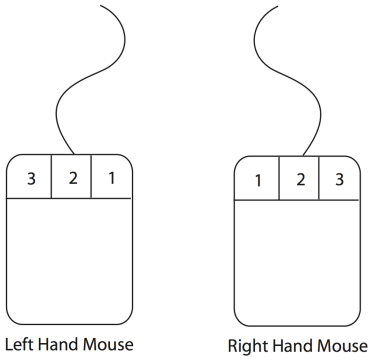


Figure 4.1: Bilateral Function Symmetry

In symPut, the functionality of the mouse buttons has been mapped with a left-right reflection symmetry. Thus, pressing the left (main) button on the right mouse is equivalent to pressing the right (main) button on the left mouse. This follows the bilateral function symmetry guideline described in Section 3.5. symPut takes input from the two mice and maps the button events to Buttons 1, 2 and 3. Thus, the left button on the left mouse and the right button on the right mouse both get mapped to Button 3, as shown in Figure 4.1. For ease of understanding, names are used for the buttons instead of numbers, so Button 1 is the *main* button, Button 2 is the *middle* button and Button 3 is the *outer* button. These names are used in the procedures above.

Each of the procedures required by symPut includes a device parameter. The symPut driver passes through a 0 if the device was generated by the left mouse and a 1 if the device was generated by the right mouse. Although it is irrelevant whether a button was pressed on the left or right mouse for most symmetric interactions, that information is needed in order to support asymmetric interactions. In addition, because the applications using symPut are responsible for drawing the two custom cursors, it is necessary to distinguish left mouse x and y movements from right mouse x and y movements.

If the mice used with symPut are three button scroll mice, mouse wheel events are also sent to the application. In addition, symPut differentiates between mouse wheel scrolling and clicking with the mouse wheel, and treats the latter as a middle button down and middle button up event series.

4.2.1 Differences under Macintosh OS

Under Macintosh OS X, the symPut driver works almost exactly the same as under Linux. The only difference is in the handling of scrolling events. Under Linux, accelerated scrolling is sensed

by the USB mouse driver and the value parameter may contain any integer. Positive integers reflect forwards scrolling and negative integers reflect backwards scrolling. Larger integers, in the absolute sense, reflect faster scrolling. Under Macintosh OS X, there is no accelerated scrolling, and so forwards scrolling always has a value of 1, while backwards scrolling always has a value of -1. The lack of acceleration can make scrolling through a large domain very time consuming.

4.2.2 Double-clicks

The symPut driver leaves the interpretation of 'click' and 'double-click' events to the application developer. The rationale for this design decision is that click and double-click speeds will be different between the two hands, especially for novice users who have not yet developed mousing skill in their non-dominant hands. Thus, it would be difficult to setup thresholds for clicking and double-clicking that would work for both hands.

4.3 symDraw

SymDraw is a prototype two-handed symmetric drawing program. In SymDraw, it is not possible to create a shape using only one mouse. For each shape in SymDraw, each mouse controls a cursor that is mapped to one of two control points on the shape. The necessity of using two mice for drawing contradicts the guidelines set out by Chatty in 1994 [Cha94b, Cha94a]; he recommends any two-handed interface allow a one-handed interaction, in case of the failure of one of the devices. However, SymDraw is specifically designed for symmetric two-handed interaction. Additionally, device failure is not a concern, because SymDraw runs with standard USB three-button wheel mice. The SymDraw prototype was written in C using the X graphics library and was then ported to use the more robust and coherent OpenGL graphics library. More recently, symDraw was ported to operate under Mac OS X.

The two cursors that are controlled by the mice differ in colour; the cursor controlled by the left mouse is red, the other cursor is blue. The cursors also differ in orientation: the cursor controlled by the left hand is an arrow pointing upward to the right, and the cursor controlled by the right hand is an arrow pointing upward to the left. This mirrors the orientation of the hands when using a keyboard (i.e. arms angled and both hands pointing forwards towards the center of the keyboard).

The symDraw application uses all three buttons on both mice in a symmetric fashion. In the rest of this description, I will refer to the inner, middle and outer buttons to describe how functions are invoked (see Section 4.2). The inner button is used for creating and manipulating objects, and for making menu selections if a menu is invoked. The middle button is used for selecting objects to edit. The outer button invokes or dismisses menus.

4.3.1 Parallel Transformations

Although computer users may not think in geometric terms, they are performing geometric transformations when they drag a file folder, resize a window, select some text or rotate an image. In current graphical user interfaces, user tasks reflect the affordances of a single spatial input device. With a single mouse or trackball, a user can perform either a translation, a scale or a rotation, but cannot easily combine any of these actions. A scale and rotation can be combined only if they both occur with respect to some previously selected anchor point.

Experienced computer users have become accustomed to the paradigm of performing simple geometric transformations in serial. However, this is unnatural compared to the way people combine translation, rotation and scaling seamlessly in everyday interactions with objects. An elastic band, for example, can be rotated, translated and scaled simultaneously. If you want to put an elastic band around an object, you use your hands to stretch the elastic band, while translating and rotating your hands together to get it in place around the object. Even when objects cannot be scaled, we effortlessly translate and rotate them at the same time. It is awkward, and sometimes impossible, to serialize these manipulations. And yet, computer users routinely manipulate on-screen objects this way.

The addition of a second spatial input device alleviates the need to switch between geometric transformations and permits transformations that combine scale, rotation and translation. The main question that arises is how to map the two degrees of freedom from each spatial device to the three geometric transformations. An asymmetric mapping might use the non-dominant hand to control the translation of an object, while the dominant hand controls scale and rotation. A symmetric mapping would involve the two hands controlling opposite corners of the object to 'steer' it through space, achieving simultaneous scale, translation and rotation. The symmetric mapping may yield the highest performance for these tasks. One of the main benefits of the symmetric mapping found in symDraw is that the user doesn't have to remember which hand is performing which task, since they are both 'steering'.

4.3.2 Interaction Symmetry

SymDraw takes into account the visual integration recommendation of Balakrishnan and Hinckley [BH00], but takes it a step further. Instead of allowing manipulation of two separate objects that are visually connected, symDraw restricts manipulation to a single object at a time. Thus, visual integration between what the two hands are working on is guaranteed. Basic shape drawing and editing are accomplished by pressing down the inner button on both mice and dragging to size and position the shape. This bimanual drawing technique allows both positioning and sizing to be performed in a single operation in a manner that is extremely fast and natural. As can be seen in Figure 4.2, both hands play a crucial and symmetric role

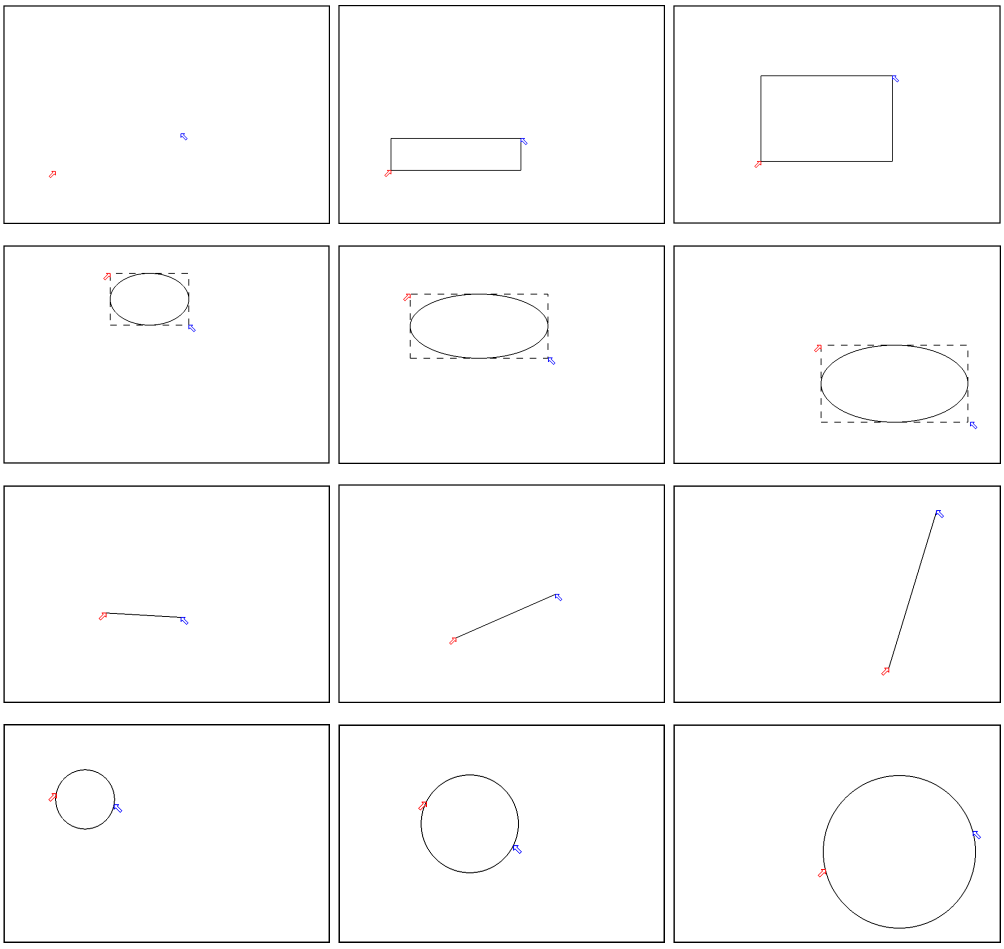


Figure 4.2: Symmetric shape creation in symDraw: rectangles, ovals, lines and circles.

in creating each shape. This forces skill development in the user's non-dominant hand. The initial SymDraw prototype allowed drawing of the following shapes:

- Rectangles: The cursors control opposite corners of the rectangle.
- Ovals: The cursors control opposite corners of the bounding box of the oval.
- Lines: The cursors control the ends of the line.
- Circles: The cursors control opposite points on the circumference of the circle.

The current version of symDraw also allows for the creation and editing of spline curves. However, spline curves are more complex than the basic geometry that is the focus of this chapter. Thus, the spline creation technique and the symmetric spline manipulation technique will be described in Chapter 5.

At any point while creating one of the basic shapes, the user can release the inner button on either mouse. This has the effect of anchoring the shape under the cursor controlled by that hand. For example, if the user releases the inner button on their right mouse while drawing a rectangle, the corner of the rectangle controlled by that cursor becomes anchored at its current position. By moving the left mouse, the user can stretch or shrink the rectangle from that point. If the user presses down the inner button on their right mouse again, the anchored corner of the rectangle moves freely again. The shape is committed only after the inner button has been released on *both* mice.

SymDraw's symmetric interaction is especially useful for quickly generating geometric design patterns such as the one shown in Figure 4.3, which was created in under 3 minutes. A design such as this, involving exact placement of circles with respect to other circles, could be very tedious to create in a standard graphics application.

4.3.3 Editing Objects

Objects that have already been drawn in symDraw can be picked and edited. Users select objects by clicking on them with the middle button on either device. The selected object is highlighted and then the user can begin editing the size, location and rotation of the object by clicking the inner button on either mouse, as illustrated in Figure 4.4. Clicking the outer button instead brings up an editing menu, which allows users to choose various other options, such as copying, moving the object backwards or forwards or changing the object's colour, see Figure 4.5.

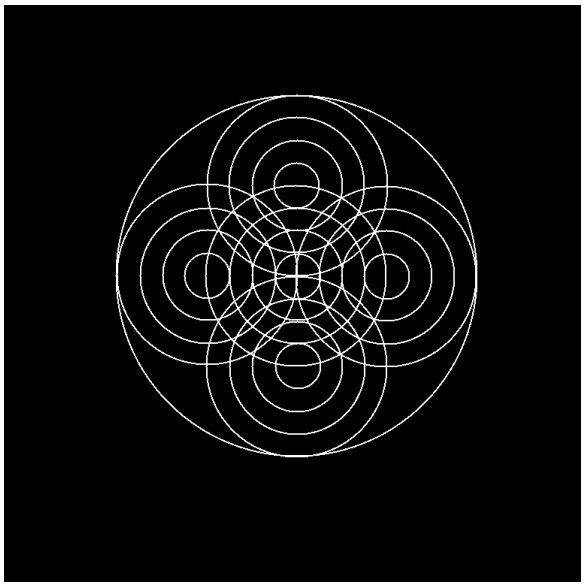


Figure 4.3: Geometric design with symDraw.

4.3.4 Transparent Menus

Transparent, hierarchical, radial menus are used in SymDraw to ensure an interface that is presentation symmetric. These menus are modeled on the ToolGlass style of menu [BSP+93, HIVB95], however, they do not use the click-through functionality that is characteristic of the toolglass interaction. That type of interaction is asymmetric and does not fit the design goals of the research. Actuation symmetry ensures that the menus can be invoked by either hand, by clicking the outer button on either mouse. Once the menu has been invoked, the mouse that invoked the menu controls the position of the menu. Thus, if the menu happens to be covering up an object of interest, the menu can be moved. Additionally, there is symmetry in the use of the menu once it has been invoked. In order to choose one of the options in the menu, the user can move the cursor (controlled by the mouse that didn't invoke the menu), or the user can move the menu under the cursor, or the user can move both at the same time. When the cursor is in position over the desired menu item, clicking the inner button on *either* mouse selects the item. To dismiss a menu without making a selection, the user clicks the outer button on either mouse. When a menu is invoked the other cursor is warped to the menu, reducing the movement necessary in order to select a menu item.

An important aspect of the transparent menus is that they are pop-up menus. That is, they don't reside in a particular place on screen, as is the case with traditional menus and toolbars. In addition to eliminating movement time to get to the menus, this design ensures

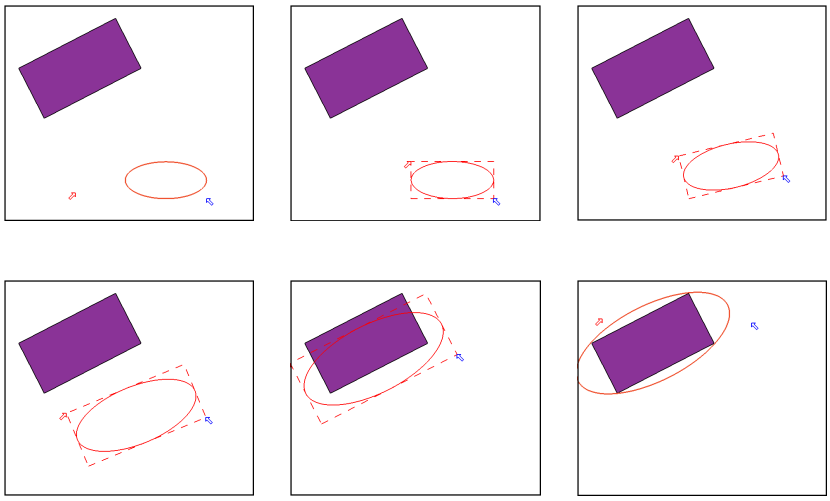


Figure 4.4: Symmetric interaction allows objects to be scaled, translated and rotated simultaneously, simplifying tasks such as enclosing a rectangle within an oval.

that the menus are not always invoked by the same mouse. As pointed out by Bier et al. , this menu style also offers the benefits of devoting more screen real estate to application data, since the menus are only displayed when needed [BSP+93]. When a sub-menu is selected the parent menu is displayed in the background in grey tones, see Figure 4.6. This helps to maintain context, so that the user knows where he is in the menu structure. In SymDraw, a menu can easily be invoked by either mouse. This encourages the use of the non-dominant hand, and can therefore help in the mousing skill development of that hand.

The design of the SymDraw menu system allows symmetric interaction: the user can invoke the menu with either mouse, can move both the cursor and the menu together, and can make a selection by clicking the inner button on either mouse. However, the task of selecting an item from a menu is a target selection task, and therefore falls under the asymmetric guidelines set out by Guiard [Gui87]. Because symmetric interaction is a superset that includes asymmetric interaction, the typical asymmetric style of menu selection is available and likely to be used. Thus, most users will activate the menu with the outer button on one mouse, leave the menu sitting where it was activated, and use the cursor controlled by the other mouse to make a selection from the menu. An interesting learning effect appears here. Because most users are accustomed to single mouse interfaces, they are likely to activate the menu using the outer button on the right mouse. This means that the cursor they will use to select a menu item is the cursor controlled by the left mouse. For right-handed users, this is counter to Guiard's description of how such interactions should work, because the right hand is doing the initial, easy task of activating the menu, and the left hand is doing the more spatially-challenging task

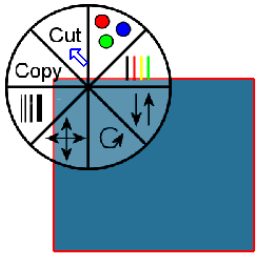


Figure 4.5: When an object is selected, it is highlighted in red, and the user can invoke an editing menu by pressing Button 3 on either mouse.

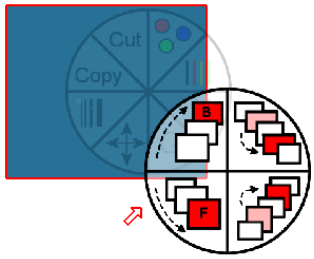


Figure 4.6: Submenus, such as this layering menu, are shown with the parent menu displayed behind in grey tones.

of selecting the menu item. However, as users become accustomed to the two-handed interface, they will acquire mousing skill in their non-dominant hand that lessens the challenge associated with this target acquisition task. They may also start activating the menu with their non-dominant hand. Alternatively, they may invoke a menu with their dominant hand and then move the menu until it is over the non-dominant hand's cursor and activate the menu selection by clicking the inner button with the dominant hand. In this way, the user can perform menu operation without using the non-dominant hand at all, if they choose.

4.3.5 Colour Selection

The colour picking technique employed in SymDraw is unique [CFC03]. The colour space that is used is the RGYB colour geometry proposed by Ware and Cowan [WC90]. The colours are presented in a square graph on which the two diagonal axes display red/green and yellow/blue

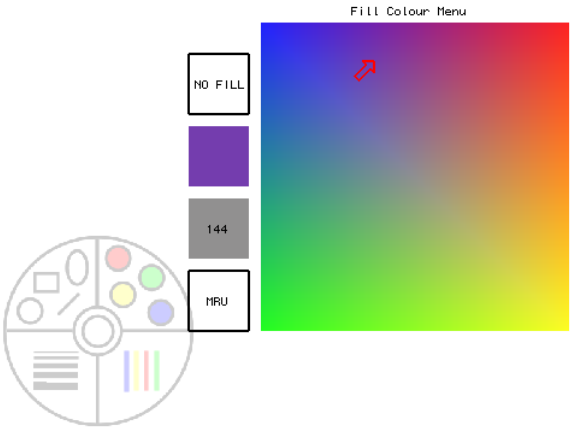


Figure 4.7: colour Selection Tool at the 144/255 luminance level.

hues. There are 255 such graphs, ranging in luminance from 0 to 255. The colour selection tool, with a middle luminance level displayed, is shown in Figure 4.7.

As with other menus, the user can invoke the colour selection tool with either mouse by selecting it from the main menu. If the colour selection menu has been invoked with the left mouse, that mouse then controls the position of the colour palette, while the right mouse controls the cursor. Rolling either mouse wheel increases/decreases the luminance level of the palette. The colour selection tool allows symmetric colour selection. The user searches for a specific colour by moving the cursor over the colour palette or moving the colour palette under the cursor or by moving both the cursor and the palette at the same time. A larger swatch of the colour currently under the cursor is shown in a block beside the colour palette. colour selection is achieved by pressing the inner button on either mouse.

Users can also pick a 'No Fill' option to make an object transparent. They can pick a grey-level option from beside the colour palette to get a pure form of white/black/grey at the current luminance level. There is also a 'Most Recently Used' option which allows users to pick from 12 recently used colours. Finally, users are able to pick a colour directly from the scene by clicking on the scene with the free cursor.

4.3.6 Wheel Scrolling

As with the other actuators used in the SymPut system, the scroll wheel on either mouse activates the same functionality in SymDraw. The functionality of the scroll wheel depends on the context, and therefore, the hand that actuates the scroll wheel may differ from task to task. The scroll wheel on current mice happens to be nicely suited for use by the non-dominant

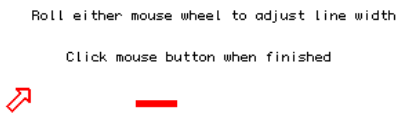


Figure 4.8: Line width is adjusted in symDraw by rolling either mouse wheel up or down.

hand. Because the wheel has physical notches that can be felt as the user scrolls, it is very easy for the non-dominant hand to scroll with precision.

In SymDraw the scroll wheel is used for four different purposes. The default action of the scroll wheel is to undo or redo shape drawing, depending on the direction of the scroll. This 'rapid undo' was very much embraced by users who tested the system. When a menu is present on the screen, the scroll wheel is used to vary the colour of the menu lines from white to black. The ability to quickly adjust the menu to achieve better contrast is important for a transparent menu system that can be floating over objects of any hue and saturation. When the colour selection palette is displayed, scrolling the mouse wheel adjusts the luminance of the colour palette. Finally, when the user selects the line width menu item, a sample line is displayed, and the user can interactively increase or decrease the line width by scrolling the wheel up or down, see Figure 4.8. In accordance with the actuation symmetry of symPut, all of these features can be activated by either hand. Users testing the system were observed to use either the dominant or non-dominant hand for the undo function, however, users seemed to prefer the dominant hand for changing line width.

4.3.7 Informal Evaluation of symDraw

Throughout the development of the symDraw application, a variety of artists were asked to try the program and provide feedback. One artist in particular was asked to test each new feature of symDraw as it was developed. This iterative testing and feedback cycle was invaluable and helped to shape the final version of the application. The main test artist provided the following comments relating to her use of symDraw and how it compares to the other computer graphics packages she regularly uses:

> After using symDraw, I find interactions with existing drawing programs rather lacking. Drawing lines, rectangles and ovals while placing and resizing them seems perfectly natural. The traditional way of drawing these objects, which requires

> guessing the right size and right location, refining multiple times in relation to other objects, seems tedious and clumsy. In Photoshop and the Gimp[1], I avoid drawing splines where possible, as I often make mistakes clicking on a handle while looking to move an edit point, and vice versa. With symDraw, there are no such mistakes. Another symDraw feature I enjoy is the colour palette. Changing values via the mouse wheel is extremely useful for me for finding just the right shade. I find myself preferring to use the right hand mouse for the menus, using symDraw on a left hander's workstation is a definite plus. The downside to symDraw is that it is a basic drawing program.. there are a lot of features that can benefit from symDraw's two handed interaction: Selection via two handed input would help editing images tremendously; I have always had trouble getting the right gradient for colours, so using two locations to preview a colour ramp would be great; Using the mouse wheel to allow you to change the shade of colour on an object would be very handy.

The other artists who tested symDraw also commented on the ease of interaction and were surprised at how easily they could manipulate the shapes using both mice. Early testing with these users showed difficulty in using marking menus based on pre-timed gestures, and this feature was removed from the menu implementation as a result of their comments. Of course, all of the artists who tested symDraw requested more features. However, the objective of this thesis was not to implement a fully-featured graphics package. The requested features would certainly be necessary in a commercial version of symDraw.

4.3.8 symDraw Summary

The symDraw application provides an example of a presentation symmetric interface. The users who tested the application were comfortable with the minimalist approach to the interface and had no trouble accessing program functionality using the transparent pop-up menus. Two of the users commented on the simplicity of the interface and how nice it was to have their image uncluttered by interface widgets. The test users all adapted quickly to the dual-mouse interaction and were pleased by the expressiveness the two mice afforded. There was some cursor confusion in the earlier versions of symDraw, which used identical, crosshair-shaped cursors (they differed in colour), but this difficulty was eliminated with the use of shape-differentiated cursors.

While the techniques for shape creation and manipulation in symDraw are not new (excepting the spline technique, which will be discussed in the next chapter), a symmetric application that affords symmetric, asymmetric and single-mouse interaction is new. By using the symPut driver, symDraw allows users to perform tasks such as menu item selection either with a single

[1] www.gimp.org

mouse, or using the two mice asymmetrically. Thus, symDraw is an example of the principle of the symmetric interaction superset, which affords asymmetric interaction.

The first technique evaluated is the one used to position and rotate objects, which involves simultaneous rotation and translation. The second technique evaluated adds in scaling, and allows a user to simultaneously rotate, translate and scale an object, provided that the aspect ratio of the object remains fixed. Neither of these techniques are new: both were briefly described by Kurtenbach et al. in their T3 system [KFBB97]. However, neither technique has been evaluated.

4.4 Translation and Rotation Evaluation

4.4.1 Previous Work in Rotation and Translation

Previously studied techniques for two-handed geometric transformations have generally followed the Guiard guidelines. The prototype rotation and scale tools described by Fitzmaurice et al. used two bricks to control transformations [FIB95]. Although the two devices were symmetric in functional capability, the techniques used for rotation and scale were asymmetric, requiring that one brick act as an anchor or point of rotation, while the brick controlled by the dominant hand adjusted angle or stretch. The authors did not indicate whether they allowed users to translate while rotating or scaling. Shaw's THRED [SG97] system also followed Guiard's guidelines for asymmetric interactions.

Many commercial graphics applications use a single mouse with modifier keys on the keyboard to allow users to switch modes between the sizing and positioning of an object. This approach is the typical time-multiplexed input technique: a single mouse can do only one spatial task at a time, and so the modifier keys serve the purpose of specifying the temporal divisions between separate spatial tasks. An even less efficient alternative is the use of toolbar buttons to switch modes. This is less attractive because two spatial sub-tasks are required to switch modes: the user has to move the mouse to the button, click and then move the mouse back to the object being manipulated. The benefit of the keyboard approach is that with single-mouse systems, the user's non-dominant hand generally rests on or near the keyboard, which makes the mode switch relatively fast compared to pressing toolbar buttons.

4.4.2 Experimental Design

The task of object positioning is common in computer applications, especially drawing and design. The symPut driver and symDraw application allow two cursors to position objects symmetrically. The important issue is whether the spatial input from a second device can

Technique	Translation	Rotation
Mouse	Drag with left mouse button pressed	Drag with right mouse button pressed
Mouse & key	Drag with left mouse button pressed	Drag with 'r' key and left mouse button pressed
Asymmetric	Drag left mouse with right button pressed	Drag right mouse with left button pressed
Symmetric	Unified steering: drag both mice with inner buttons pressed	

Table 4.1: Techniques tested in rotation-translation experiment.

be used to make the task of rotating and translating objects more efficient. The experiment presented here aims to answer that question by comparing four techniques (described in Table 4.1): the symmetric technique used in symDraw (similar to the 'two-handed stretchies' described by Kurtenbach et al [KFBB97]), a dual-mouse asymmetric technique in which the non-dominant hand controls translation of the object, while the dominant hand rotates the object, and two single-mouse techniques in which the mode switch between translating and rotating is achieved by pressing a keyboard modifier or pressing a second mouse button.

An alternative technique, in which a single mouse is used to simultaneously rotate and translate an object using a pseudo-physics algorithm [BL01, KCST05], was not considered. That type of technique does not offer the same type of precision as the techniques tested, although a comparison would be interesting.

Task

The experimental task consisted of the translation and rotation of a right-angled triangle onto a target triangle. The single-mouse techniques chosen are standard techniques from current software. The dual-mouse asymmetric technique involves using the non-dominant hand to control translation of the object, while the dominant hand controls rotation of the object. Becuase the object rotates around its center, the non-dominant hand sets the frame of reference in which the dominant hand works, by controlling the location of the object on screen.

In each experiment trial, the participant was asked to position a green triangle over top of an identically-sized red triangle which was in a different location and orientation on screen (Figure 4.9). Each trial began with the appearance of one or two small start circles. The participants had to move their cursor(s) into the start circles. This was followed by four 'readiness' beeps. Then the triangles would appear and the trial timing would start. In each trial, the red target was the same distance from the green triangle, but in a variety of directions. In addition, the red target triangle was rotated around its center by some angle. The angles of

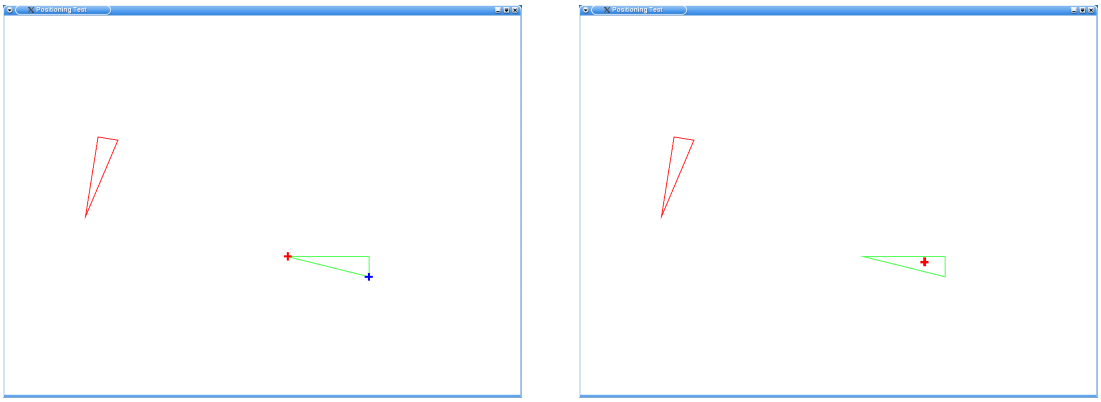


Figure 4.9: Participants were asked to move the green triangle on top of the red triangle. The left image shows the two cursors used in the symmetric technique, the right image shows the single cursor used in the other techniques.

rotation were ±30, ±60, ±120 and ±150. Thus, each trial varied in the direction of translation and in the amount of rotation required to align the green triangle with the red target triangle.

4.4.3 Stimulus-Response Incompatibility

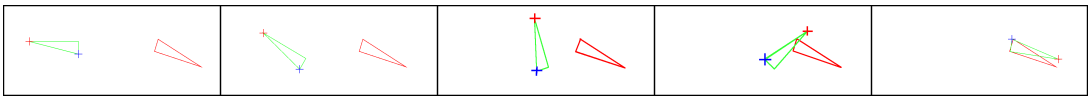


Figure 4.10: A series of experiment screenshots illustrating the cursors crossing one another during a symmetric trial. The red cursor is controlled by the left mouse and the blue cursor is controlled by the right mouse.

In pilot testing of the experiment, an interesting issue arose with the symmetric technique. Whenever the triangle was turned more than ±90°, the cursors flipped. Initially, the cursor controlled by the right mouse is located on the right side of the triangle and the cursor controlled by the left mouse is on the left, but as the triangle is rotated through vertical, the cursors cross over (Figure 4.10). Some pilot participants found this to be confusing initially. This problem is well-known in psychology as a stimulus-response compatibility issue [FD54]: the stimulus of moving the right mouse is not compatible with the response of the movement of the cursor farthest to the left on screen. Although I could have avoided this problem by changing to a symmetric shape such as a rectangle (which would never have to be rotated more than ±90°), I considered it important to determine how much impact the stimulus-response compatibility issue would have on users.

4.4.4 Main Experiment

16 undergraduate (non-computer-science) students were participants for the first experiment. Eleven of the students were female, five were male. All participants self-identified as right-handed. The experiment session was a repeated-measures design, consisting of four conditions, one for each technique described in Table 4.1. For each condition, there were ten practice trials, followed by 40 randomized trials. Thus, each participant completed $4 \times (10 + 40) = 200$ trials in total, and 160 timed trials. The order of the four conditions was randomized across participants. Each experimental session lasted roughly 40 minutes and participants were paid CDN$10 for their participation.

There were $16 \times 160 = 2560$ timed trials to analyze. Three out of the 2560 trials were thrown out because the participants did not complete them. In these cases the participants moved the mouse or mice so fast that they lost the triangle and could not reacquire it. The results across the remaining 2557 trials are summarized in the second and third columns of Table 4.2.

When all 2557 trials are analyzed together, the two single mouse techniques are faster than the two dual mouse techniques (Lines 1-4 in Table 4.2). While a repeated measures analysis of variance (RM ANOVA) shows that the condition variable is statistically significant (F-ratio = 7.74, $p_{3,2498} \leq 0.0001$), LSD post-hoc tests show that the difference between the symmetric techniques and the other three techniques is not statistically significant (Lines 4-6 of Table 4.3). Note that techniques whose difference shows a probability less than 0.05 in Table 4.3 are not considered to be statistically different.

When the mean trial times are separated between trials in which the angle of rotation was less than 90° in either direction, and trials with rotations greater than ±90°, the results are different for the symmetric technique. For small rotations, the symmetric technique is clearly faster than the other three techniques (Lines 5-8 of Table 4.2). The difference between the symmetric techniques and the other three techniques for small rotations is statistically significant (Lines 10-12 of Table 4.3). For large rotations, the symmetric technique is clearly much slower than the other techniques (Lines 9-12 of Table 4.2). The difference between the symmetric technique and the other three techniques is again significant (Lines 16-18 of Table 4.3). The significantly slower trial times for the large rotation trials in the symmetric condition is likely a result of the stimulus-response compatibility issue described earlier.

The results support the idea that the crossing cursors cause confusion and increase completion time. However, this does not mean that the symmetric interaction technique should be dismissed. For small rotations, the symmetric technique is 18% faster than the next fastest technique (mouse & key). Of course, the symmetric technique is the slowest technique for large rotations. There are no data to suggest that large rotations are less common than small rotations in modeling tasks. However, it seems likely that small rotations would be more common simply because any symmetric object would not require rotations beyond a magnitude

	All Trials		
	Technique	Mean Time	Std Dev
1	Mouse	4.20	2.11
2	Mouse & key	4.21	2.23
3	Asymmetric	4.82	2.42
4	Symmetric	4.45	4.54
	ANOVA F-ratio for Condition variable is 7.74, $p_{3,2498} \leq 0.0001$		

	Small Rotation Trials		
	Technique	Mean Time	Std Dev
5	Mouse	4.02	2.17
6	Mouse & key	3.90	1.90
7	Asymmetric	4.88	2.48
8	Symmetric	3.29	1.96
	ANOVA F-ratio for Condition variable is 50.95, $p_{3,1493} \leq 0.0001$		

	Large Rotation Trials		
	Technique	Mean Time	Std Dev
9	Mouse	4.46	2.01
10	Mouse & key	4.68	2.50
11	Asymmetric	4.74	2.32
12	Symmetric	6.22	6.40
	ANOVA F-ratio for Condition variable is 14.39, $p_{3,987} \leq 0.0001$		

Table 4.2: Main experiment, first 40 minutes of exposure to task and techniques. Results (in seconds) over all trials, trials with small rotations, and trials with large rotations.

of 90°. If this is the case, then the symmetric technique's advantage is even more significant than these results suggest. Experiment 2 was designed to further examine how users perform large rotations with the different techniques over time.

4.4.5 Followup Experiment

In the second experiment on rotation and translation, six participants were randomly recruited from the 16 participants who participated in the main experiment. These six participants were asked to return and perform the same experiment three more times over a period of four days. The purpose of the followup experiment was to examine the change in the completion time of trials as the participants became more accustomed to the task and the techniques. The expectation for this experiment was that, over the course of four experiment sessions, the symmetric technique would emerge as the most efficient, as users became accustomed to the

LSD Post Hoc Tests, All Trials			
	Techniques	Difference	Prob
1	Mouse & key - Mouse	0.013169	0.929520
2	Asymmetric - Mouse	0.625735	0.000027
3	Asymmetric - Mouse & key	0.612566	0.000040
4	Symmetric - Mouse	0.262371	0.078502
5	Symmetric - Mouse & key	0.249202	0.094554
6	Symmetric - Asymmetric	-0.363364	0.014810
LSD Post Hoc Tests, Small Rotation Trials			
	Techniques	Difference	Prob
7	Mouse & key - Mouse	-0.127481	0.327628
8	Asymmetric - Mouse	0.853855	$< 10^{-6}$
9	Asymmetric - Mouse & key	0.981336	$< 10^{-6}$
10	Symmetric - Mouse	-0.740544	$< 10^{-6}$
11	Symmetric - Mouse & key	-0.613063	0.000003
12	Symmetric - Asymmetric	-1.594398	$< 10^{-6}$
LSD Post Hoc Tests, Large Rotation Trials			
	Techniques	Difference	Prob
13	Mouse & key - Mouse	0.223607	0.462849
14	Asymmetric - Mouse	0.283018	0.352811
15	Asymmetric - Mouse & key	0.059410	0.845325
16	Symmetric - Mouse	1.791536	$< 10^{-6}$
17	Symmetric - Mouse & key	1.567929	$< 10^{-6}$
18	Symmetric - Asymmetric	1.508519	0.000001

Table 4.3: Main experiment, LSD post hoc tests for condition variable for all trials, for small rotation trials only and for large rotation trials only. Differences in seconds.

cursors flipping. The results across all trials are summarized in the third and fourth columns of Table 4.4.

The results in Lines 4, 8 and 12 of Table 4.4 show that the total completion times for the task using the symmetric technique improve dramatically over only a few sessions. The total completion time also improved significantly for the dual mouse asymmetric technique, indicating that dual mouse interfaces require some learning. However, the symmetric technique is faster than the asymmetric technique in both experiments and this seems natural given the symmetric nature of the task. The completion time improvement for the symmetric technique was evident for both the trials with small rotations and the trials where the rotation was large enough to cause the cursors to flip. Lines 4-6 of Table 4.5 show that the symmetric technique is

	All Trials		
	Technique	Mean Time	Std Dev
1	Mouse	4.08	3.15
2	Mouse & key	3.91	2.70
3	Asymmetric	4.10	3.03
4	Symmetric	3.61	2.46
	ANOVA F-ratio for Condition variable is 12.2, $p_{3,3792} \leq 0.0001$		

	Small Rotation Trials		
	Technique	Std Dev	Mean Time
5	Mouse	4.04	3.50
6	Mouse & key	3.77	2.64
7	Asymmetric	4.14	3.16
8	Symmetric	3.13	1.72
	ANOVA F-ratio for Condition variable is 29.53, $p_{3,2272} \leq 0.0001$		

	Large Rotation Trials		
	Technique	Mean Time	Std Dev
9	Mouse	4.15	2.54
10	Mouse & key	4.13	2.77
11	Asymmetric	4.06	2.82
12	Symmetric	4.34	3.14
	ANOVA F-ratio for Condition variable is 1.33, $p_{3,1512} = 0.2629$		

Table 4.4: Followup experiment, two hours of exposure to task and techniques. Results (in seconds) over all trials, trials with small rotations, and trials with large rotations.

significantly faster than the other three techniques when all trials are considered together. The same is true for small rotation trials (Lines 10-12 of Table 4.5). The difference in performance for the four techniques is not significant for large rotations (Lines 13-18 of Table 4.5). It seems likely that participants' ability to deal with the crossed cursors is due to the linkage of the cursors to an object, which the users understand to be flipped over. However, this is an area that needs more detailed examination.

The Gender Gap

An unexpected result of the analysis is the significant effect of completion time for gender. In both experiments, the completion times for males were significantly lower than those of the female subjects. The standard deviation for the male subjects was low. For female subjects, the mean completion time was high, and the standard deviation was high (Table 4.6). The

LSD Post Hoc Tests, All Trials			
	Techniques	Difference	Prob
1	Mouse & key - Mouse	-0.172671	0.062049
2	Asymmetric - Mouse	0.023443	0.799961
3	Asymmetric - Mouse & key	0.196114	0.034078
4	Symmetric - Mouse	-0.472061	$< 10^{-6}$
5	Symmetric - Mouse & key	-0.299391	0.001221
6	Symmetric - Asymmetric	-0.495505	$< 10^{-6}$
LSD Post Hoc Tests, Small Rotation Trials			
	Techniques	Difference	Prob
7	Mouse & key - Mouse	-0.274319	0.020402
8	Asymmetric - Mouse	0.097030	0.411849
9	Asymmetric - Mouse & key	0.371349	0.001704
10	Symmetric - Mouse	-0.909383	$< 10^{-6}$
11	Symmetric - Mouse & key	-0.635064	$< 10^{-6}$
12	Symmetric - Asymmetric	-0.1006414	$< 10^{-6}$
LSD Post Hoc Tests, Large Rotation Trials			
	Techniques	Difference	Prob
13	Mouse & key - Mouse	-0.020198	0.889454
14	Asymmetric - Mouse	-0.086936	0.549691
15	Asymmetric - Mouse & key	-0.066738	0.646057
16	Symmetric - Mouse	0.183921	0.205753
17	Symmetric - Mouse & key	0.204120	0.160258
18	Symmetric - Asymmetric	0.270858	0.062483

Table 4.5: Followup experiment, LSD post hoc tests for condition variable for all trials, for small rotation trials only and for large rotation trials only. Differences in seconds.

results are statistically significant, generating high F-ratios and low p-values in the ANOVA. These numbers indicate that some of the female subjects were probably fast, but a number of them were slow. Such standard deviations suggest that the female subjects were at earlier points on the learning curve for the dual mouse techniques, and that it is possible that the symDrive results might still improve substantially with just a few more sessions for the female subjects. The small standard deviation calculated for the male subjects indicate that they are likely much further along the learning curve for two-mouse input and the completion times for the dual mouse techniques would not likely change much with successive sessions. The reason behind the gender difference is unclear, although it is possible that the male subjects have much more exposure to two-handed electronic interaction through the use of bimanual video game controllers. However, data on computer and video game usage were not collected, and

Experiment	Gender (#)	Mean Time	Std Dev
Initial experiment	males (5)	3.40	1.5
Initial experiment	females (11)	5.42	3.3
Followup experiment	males (3)	2.44	0.8
Followup experiment	females (3)	4.89	3.3

Table 4.6: Initial and followup experiments, mean completion times by gender, averaged over all four techniques

so that hypothesis could not be tested.

Subjective Evaluation

After the first experiment, the 16 subjects were asked to rank the 4 techniques in order of preference. Specifically they were asked "If you had to spend your life moving triangles around a screen, which method would you want to use most?". Because the subjects were not informed of their performance with any of the techniques, the objective results were not taken into consideration by subjects answering this question. The results were interesting – all four of the techniques were favoured by some of the subjects, and there was no clear winner in the subjective rankings. The same question was asked again of the six subjects participating in the followup experiment, and again, the preferred method varied with no conclusive results.

4.4.6 Rotation and Translation Experiment Conclusions

These two experiments demonstrate that the symmetric dual-mouse technique is faster than both the asymmetric and single-mouse techniques for rotation and translation of objects. Users are able to perform the symmetric task with great speed for rotations of less than 90°, faster than with any of the other techniques tested. Although users experienced some initial difficulty using the symmetric technique for trials requiring larger rotations, due to the flipping of the cursors, the follow-up experiments showed that this difficulty disappeared with repeated exposure. After approximately two hours of use, the symmetric technique was the fastest of the four techniques overall.

4.5 Rotation, Translation and Scale Evaluation

In addition to simultaneous rotation and translation, four degrees of freedom from two spatial input devices can be used to simultaneously rotate, translate and scale an object, provided that the aspect ratio of the object is fixed. Consider a straight line that stretches between

the two cursors. The length of this line is used to determine the size of the object. The angle of the line with respect to the horizontal axis is used to determine the rotation angle of the object. The position of the line on screen represents the position of the object. Thus, by moving the two cursors around the screen, the user changes the position, angle and size of the line stretching between the two cursors and that provides into a new position, orientation and size for the object. This technique is used in symDraw for editing objects that have already been created and is shown in Figure 4.4.

In order to evaluate this symmetric technique for rotation, translation and scale, it has been applied to the task domain of aligning digital images. Pasting together separate digital photographs into a panoramic image in a typical graphics program is difficult because the images often need to be slightly scaled, slightly rotated and slightly overlapped. The manual image alignment task is an appropriate choice for evaluating rotation, translation and scale techniques because it is a task that should be easy for participants to understand, and so the learning effects should be minimal.

4.5.1 The Image Alignment Task

Image registration is commonly used in photography, digital art, medical imaging and remote sensing. In image registration, two or more images must be aligned to form a single complete image. In certain domains, such as remote sensing, there is no user interaction, and the images are processed by specialized software that expects a certain ordering and specific camera angles. However, for photographers and artists, the alignment of images is often done manually. In some software, algorithms are used after manual alignment to perfect the registration and adjust factors such as lighting and shadows. For evaluating rotation, translation and scale techniques, it is necessary to concentrate on evaluating the interactive aspect of image alignment.

There is little published research in the area of manual image alignment; the research in this area has focused on automatic image registration algorithms [CK99, Sze04]. Instead, one must look to commercially available software to see what manual techniques are used. These techniques fall into two categories: manual movement of the images on a virtual desktop (such as in Photoshop and PTgui [Ser]), and point-registration methods (such as in Panavue [Com]), where the user selects coincident points on the images to be aligned.

It is worth examining how the manual image alignment task is done outside of the virtual workspace. An informal experiment, in which users were asked to perform the task with different types of paper, was conducted. Small, light-weight images were translated and rotated by one hand. For larger and/or heavier images, the image was roughly translated into place with each hand holding an edge or corner and then simultaneously translated and rotated into final position using both hands symmetrically. There are two important points to be gleaned

from this experiment. When both hands are used, they are used symmetrically. And, regardless of the size of the picture, the fine-tuning part of the interaction involved *simultaneous* rotation and translation.

A single hand can simultaneously rotate and translate a small object such as a photograph in three dimensions. In the virtual task, a single mouse provides only two degrees of freedom for translation in a two-dimensional space. Thus, simultaneous rotation and translation of the image is theoretically not possible using only one mouse, though some pseudo-physics algorithms have been proposed [BL01, KCST05]. Specialized devices such as digital pucks could provide affordances similar to the hand-wrist combination, but the research for this thesis is focused on what can be accomplished with standard mice.

The second interesting difference between the physical alignment task and the corresponding virtual task is that the virtual task allows the scaling of images, which is not possible in the physical realm. If the ability to scale images in the virtual image alignment task is included, the single-mouse pseudo-physics algorithms for simultaneous translation and rotation are insufficient. The second mouse is necessary to achieve the desired effect.

Physical image alignment demonstrates that virtual image alignment is suited to symmetric interaction. Additionally, scaling is a task that is naturally symmetric: the distance between the cursors can be used to control scale. Thus, the symmetric image alignment technique, where the two cursors are attached to opposite corners of the image, should outperform single-mouse techniques and dual-mouse asymmetric techniques.

4.5.2 Experimental Design

I designed an experiment to evaluate the symmetric rotation, translation and scale technique against a dual-cursor asymmetric technique and a single-cursor technique. The experiment was conducted as a within-participants comparison of three conditions: single-mouse, dual-mouse asymmetric and dual-mouse symmetric. 24 participants, all undergraduate students not majoring in computer science or computer engineering, were paid CDN$10 to complete the experiment. 13 participants were male, 11 were female and all self-declared as right-handed. For each condition, four images were multiplied by ten transformation configurations to produce 40 different trials which were presented in random order. Each participant completed four practice trials plus 40 measured trials in each of the three conditions, for a total of 132 trials. The three conditions allowed six different presentation orders, and the participants were grouped and counterbalanced so that each of the presentation orders was seen by four participants.

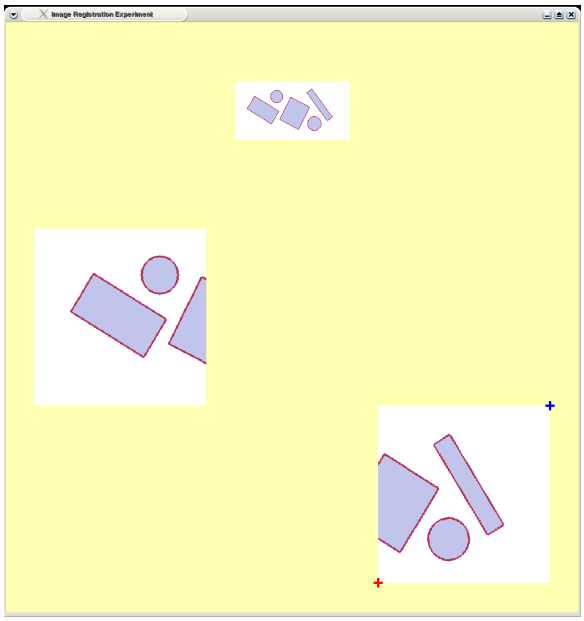


Figure 4.11: At the start of each trial the two images to be aligned are displayed, as well as a small complete image at the top of the screen, so that participants know what the final picture should look like.

Task

In each experiment trial, the participant was asked to align two images into a single, complete picture. Each trial began with the appearance of one or two small start circles. The participants had to move their cursor(s) into the start circles. This was followed by four 'readiness' beeps. Then the images would appear and the trial timing would start (Figure 4.11). Image A was always on the left side of the screen, and could not be moved. Image B was initially located in the bottom right of the screen. The participants were required to translate, rotate and scale Image B until it was aligned with Image A. Image A displayed exactly one half of the source image and was not rotated or scaled. Image B was jittered according to one of ten transformation configurations described later in this section. Image C, a small version of the full source image, was displayed at the top center of the screen, to give participants an idea of how the aligned images should look. To aid in the registration process, the edges of Image B faded to transparency while it was being manipulated. This allowed participants to see both Image B and part of Image A underneath it (Figure 4.12). When the participant aligned Image B to Image A, a red border appeared around Image A to signify that the alignment was complete (Figure 4.17). After 200 milliseconds, the images disappeared and the trial was over.

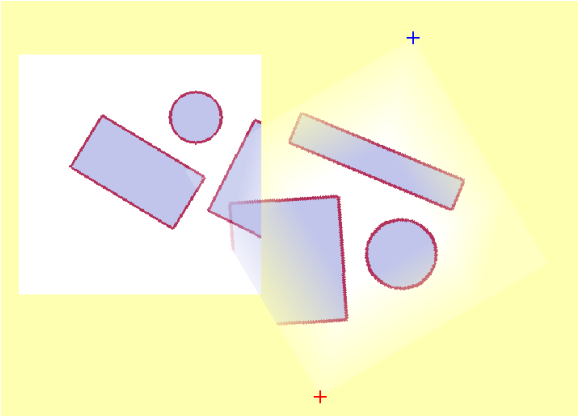


Figure 4.12: As Image B (right) is manipulated, its edges become transparent to help in the overlapping aspect of the image alignment task.

Image Sets

Four different source images were used for generating image sets and each image was seen 33 times by each participant: three conditions multiplied by ten transformation configurations, plus once in a practice trial for each condition. Simple graphics were used instead of real photographs so that participants would not be distracted by photographic details. A variety of simple graphic images were tried during pilot testing and the four images displayed in Figure 4.13 were chosen. Images 1, 2 and 3 are all straightforward to align. Image 4 is more difficult: the images can be aligned in a variety of configurations such that the rotated oval appears 'complete'. Other graphics tested in the pilot phase exhibited similar properties. Although it was desirable to see how the different techniques would fare with 'easy-to-align' images versus 'difficult-to-align' images, only one 'difficult-to-align' source image was used. This was done to keep the complete experiment session time under one hour and to minimize participant frustration.

Transformation Configurations

In a typical virtual image alignment situation, the images are similar in alignment. If one of the images is rotated 90, 180 or 270 degrees with respect to the other, there are usually buttons or menu operations that can automatically perform these large rotations. For this reason, only slight rotations (10 degrees or less) were used. Similarly, images to be aligned are usually of similar, though not identical scale. Finally, there is usually some overlap between the images, requiring one image to be translated over the other image slightly. These small 'jitter' factors are meant to reflect the slight imperfections one might see in a set of digital images taken

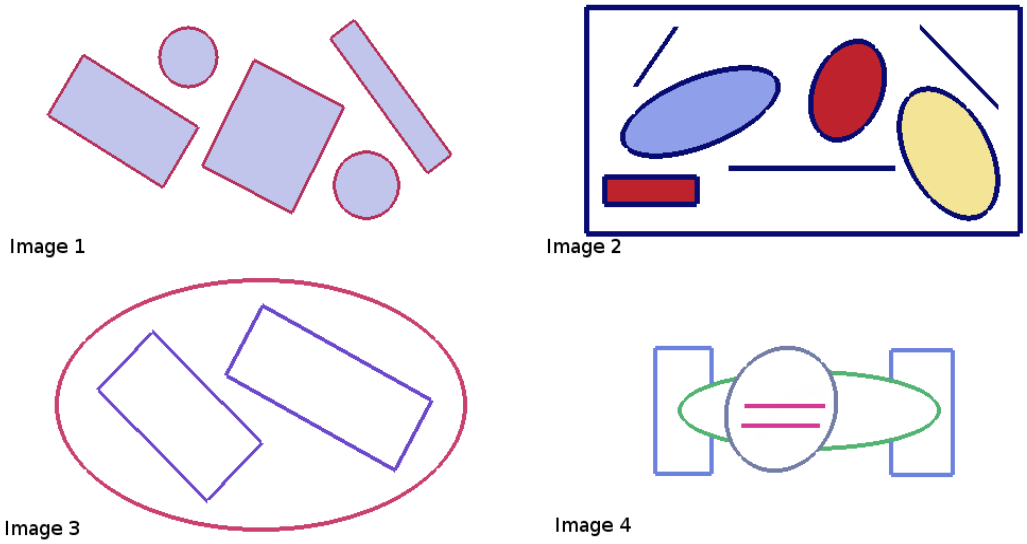


Figure 4.13: The four graphics images that were used in the experiment. Each graphic was treated as an OpenGL texture and transformed onto two rectangles to create image pairs that participants were asked to align.

by an amateur photographer trying to capture a panoramic scene. For simplicity, non-linear perspective transformations were not included. The ten transformation configurations are listed in Table 4.7, along with a partial image for each configuration.

Conditions

Three conditions were tested in this experiment: single-mouse (SING), dual-mouse asymmetric (ASYM) and dual-mouse symmetric (SYM). Each condition represents a different image manipulation technique (Table 4.8). For all three conditions, the cursor or cursors were locked to Image B.

SING Condition

The single mouse technique is similar to the image dragging technique in PTgui and Photoshop. Using the dominant hand, the participant can press down on the main mouse button and drag the mouse to translate the image. Pressing the right mouse button instead allows the participant to drag the corner of the image (this differs slightly from the commercial software where the mode-switch to rotation is activated by clicking on a toolbar button). Dragging away from or towards the center of the image scales the image up or down, maintaining the aspect

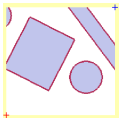
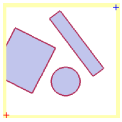
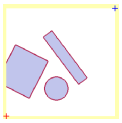
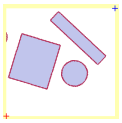
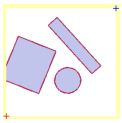
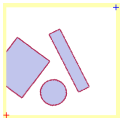
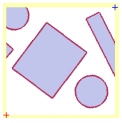
T	S	R	Image	T	S	R	Image
0.0	1.2	0		0.0	1.1	0	
-0.1	0.9	0		-0.2	0.8	0	
-0.15	1.0	10		-0.1	1.0	5	
0.0	1.0	-5		0.0	1.0	-10	
0.0	1.2	-10		-0.1	0.9	-5	

Table 4.7: Transformation configurations used in the experiment and displayed for one of the partial images. T is x-axis translation factor, S is scale factor and R is rotation factor in degrees.

ratio (Figure 4.15). Dragging in a circular motion around the center of the image rotates the image (Figure 4.16).

ASYM Condition

There are no descriptions of dual-mouse asymmetric techniques for image manipulation described in the HCI or image registration literature. Four degrees of freedom are required for the image alignment task: one for translation in x, one for translation in y, one for rotation around a fixed point and one for uniform scale. As each mouse offers two degrees of spatial input, these four sub-tasks must be split across the two mice. It is clearly inappropriate to split the two degrees of translation across the two mice. Therefore, the only sensible way

Condition	Image Translation	Image Rotation and Scale
SING	Drag with inner button pressed	Drag with outer button pressed
ASYM	Drag left mouse with inner button pressed	Drag right mouse with inner button pressed
SYM	Unified manipulation: drag both mice with inner buttons pressed	

Table 4.8: Interaction techniques tested in the image registration experiment.

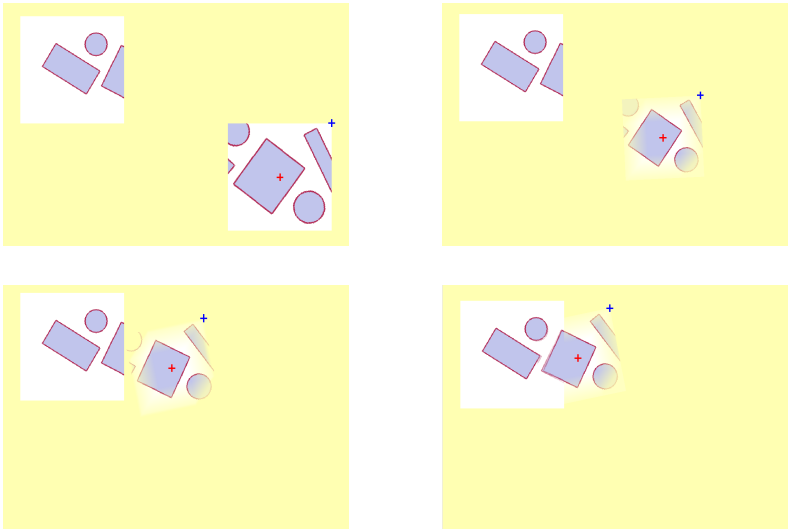


Figure 4.14: In the ASYM condition, the image can be translated by the non-dominant hand (red cursor) while being scaled and rotated by the dominant hand (blue cursor).

to split the four sub-tasks across the two mice is to have one mouse control both x and y translation, and have the other mouse control rotation and scale. The asymmetric technique chosen separates the tasks such that the non-dominant hand (using a red cursor attached to the center of Image B) controls the translation of the image while the dominant hand (using a blue cursor attached to the upper-right corner of Image B) controls the rotation and scale of the image. This separation of tasks best conforms to Guiard's guidelines for asymmetric interaction [Gui87]. In order for the rotation and scale to be performed by a single mouse, they both must occur with respect to the same point, which in this case is the center of the image. Because the non-dominant hand is controlling the position of the image, the scale and rotation of the image performed by the dominant hand is done in the frame of reference set by the non-dominant hand. Note that the separation into sub-tasks is identical in the SING and ASYM conditions: translation is one sub-task and rotation/scale is the other sub-task.

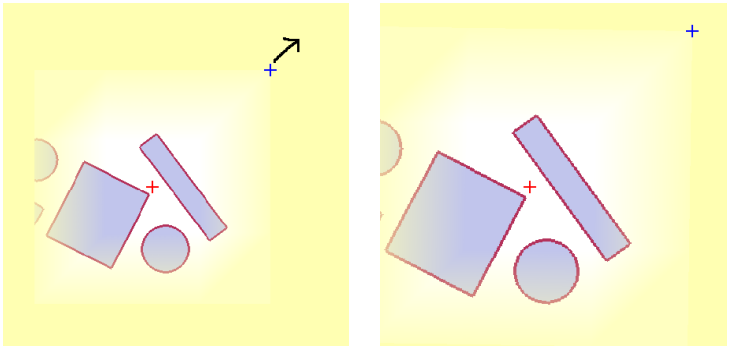


Figure 4.15: In the ASYM and SING conditions, the right cursor moves towards or away from the center of the image to scale it.

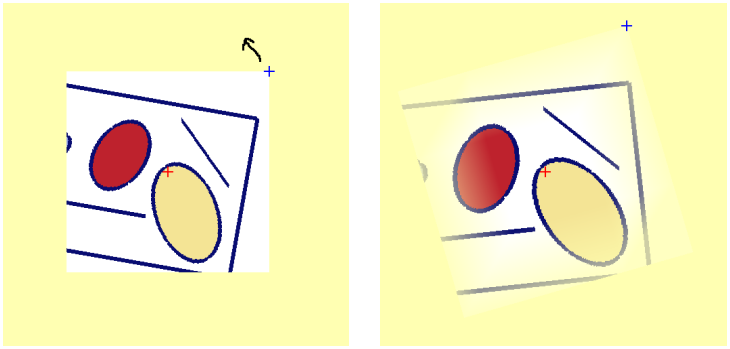


Figure 4.16: In the ASYM and SING conditions, the right cursor moves in an arc around the center of the image to rotate it.

For scale, the dominant hand must move the cursor away from the center of the image to make the image larger or move the cursor closer to the center of the image to make the image smaller (Figure 4.15). The aspect ratio of the image is maintained as the image is scaled. For rotation, the dominant hand must drag the corner of the image in an arc around the center of the image, and the image rotates in the direction of the drag (Figure 4.16). Finally, because the two mice can be moved simultaneously, the image can be translated by the non-dominant hand while being rotated and scaled by the dominant hand (Figure 4.14).

SYM Condition

The symmetric interaction technique for image alignment is simple and follows the symDraw technique described earlier in this chapter (Section 4.3.3). The cursors are attached to opposite corners of the image. By moving both cursors, the image can be rotated, scaled and translated

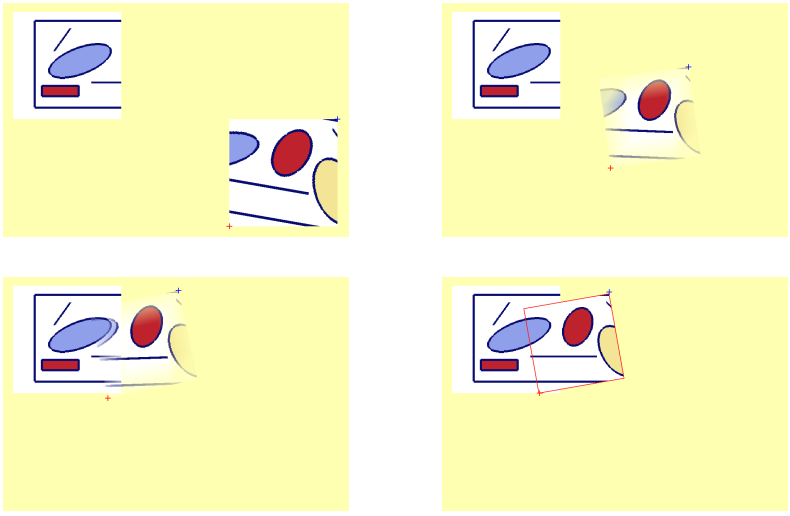


Figure 4.17: In the SYM condition, the image can be scaled, translated and rotated in a single fluid motion by moving both cursors around the screen simultaneously.

simultaneously (Figure 4.17). For this to work, the initial aspect ratio of the image is calculated. Once the cursors start moving, the diagonal line between them is used to determine the scale of the image, and the position of the two cursors determines both the location and orientation of the image.

Performance Metrics

The metrics measured were trial completion time, reaction time and movement time. Additionally, the time spent mode switching for the single-mouse technique was measured. These measurements were calculated as follows:

- *Completion Time* was measured from the time the images were first presented until the images aligned correctly.
- *Response Time* was measured from the time the images were first presented until the first mouse movement.
- *Switch Time* for the SING condition was the accumulation of time periods measured from the time either mouse button was released until either mouse button was pressed.
- *Movement Time* was measured as completion time minus response time minus switch time (switch time only applies to SING condition).

Techniques	Difference	Prob
SING - ASYM	4.489701	$< 10^{-6}$
SYM - ASYM	-4.028438	$< 10^{-6}$
SYM - SING	-8.518139	$< 10^{-6}$

Table 4.9: Image registration experiment: LSD post hoc tests for the condition variable. Differences are in seconds.

In addition to performance times for each trial, the amount of parallelism exhibited by users in the SYM and ASYM techniques was measured. The participants were considered to be using the two hands in parallel when both mice were moving continuously with the inner mouse buttons pressed.

Hypotheses

There were three hypotheses for this experiment:

- H1: The mean completion time for SYM will be shorter than the mean completion time for ASYM which will be shorter than the mean completion time for SING.
- H2: The mean SYM completion time will be shorter than the mean SING completion time, even after removal of mode switch times from the SING trials.
- H3: The level of parallelism for the SYM condition will be higher than for the ASYM condition.

4.5.3 Results

An analysis of the timing results for all trials across all participants shows evidence to statistically support all three hypotheses. Practice trials were not included, so a total of 24 participants $\times$ 3 conditions $\times$ 4 images $\times$ 10 transformation configurations = 2880 trials were analyzed.

Mean Completion Times

Figure 4.18 shows the mean trial completion time under each condition as the height of each column. The mean completion time for SYM is 9.7 seconds, approximately 87% faster than for SING (18.3 seconds). The mean completion time for ASYM is 13.8 seconds or approximately 41% faster than the SING technique. This supports H1: participants performed the task faster

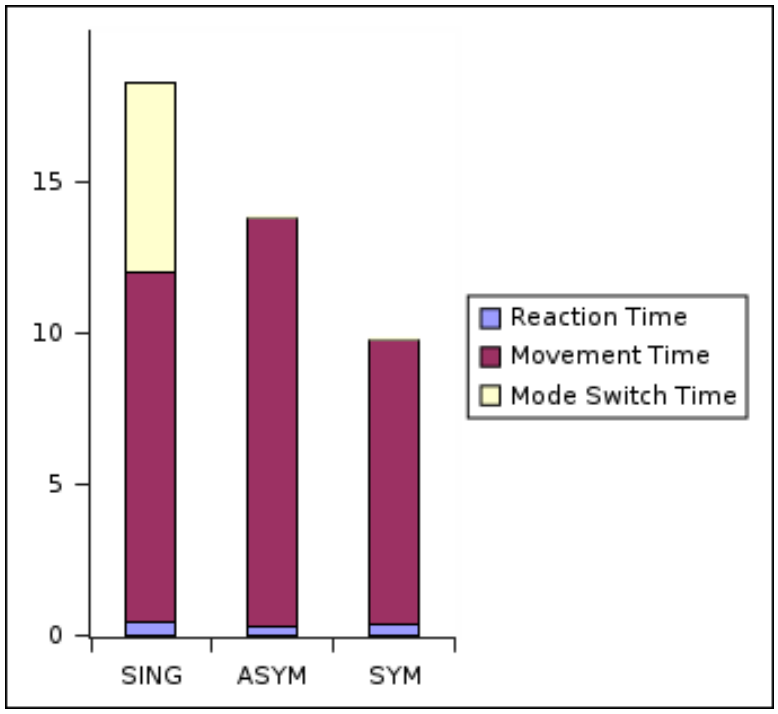


Figure 4.18: Image registration experiment: Mean trial completion times for each condition broken down vertically by reaction time, movement time and mode-switch time.

under the dual-mouse conditions than under the single-mouse condition, and faster under the SYM condition than under the ASYM condition. An RM ANOVA for trial completion time gives an F-ratio for condition of 120.54 ($p_{2,1908} \leq 0.0001$). Additionally, LSD post hoc tests were done to measure the statistical significance of the individual differences listed in Table 4.18. The results of the LSD test are shown in Table 4.9 and show that the symmetric condition is significantly faster than the other conditions.

Cognitive Load Analysis

With the SING technique, time must be spent switching modes. This time was measured from the time one mouse button was released until another mouse button was pressed. When this mode-switching time is removed from the completion times for the SING condition, the movement time can be compared to the SYM and ASYM conditions. The mean movement times for the three conditions are shown as the purple sections of each column in Figure 4.18. Note also from Figure 4.18 that reaction time did not differ significantly across the three conditions. The mean movement times support H2: the SYM condition is significantly faster

than the SING condition even after mode-switch times are removed. The RM ANOVA for movement time gives an F-ratio for condition of 59.50, ($p_{2,1908} \leq 0.0001$), showing that these results are statistically significant. The difference in mean movement times suggests that the shorter time seen in the symmetric technique is not simply because there is extra time taken in the single-mouse technique for mode switches. The mode switching takes an average of 6.3 seconds for each trial. However, there is still a 2.2 second difference in the movement times of the SYM condition and the SING condition. As well, there is a 4.1 second difference in the movement times of the SYM condition and the SING condition. These times suggest that the unified task representation in the SYM condition imposes a lower cognitive load than the other two techniques.

With mode-switching time removed, the SING condition is faster than the ASYM condition. Considering that the image registration task is split into the same sub-tasks in the SING and ASYM conditions (translation is one sub-task, and scale-rotation is the other sub-task), some other factor is producing statistically different average movement times (13.47 seconds versus 11.58 seconds) after the mode-switch time is removed. It is likely that the split into sub-tasks is a contributing factor for both conditions: SYM is 4.1 seconds faster than ASYM and 2.2 seconds faster than SING. Given these numbers, there is an approximate 2 second cost associated with the separation of the task into two sub-tasks. Thus, the ability to rotate, scale and translate simultaneously saves 2 seconds per trial in the symmetric technique. However, there is still an extra 2 second cost unaccounted for in the ASYM technique compared to the SING technique. This may be an additional cognitive burden related to remembering which hand is doing which task. This is not an issue with the SYM condition, because the two hands have exactly the same role.

Parallelism Analysis

The amount of parallel interaction between the SYM and ASYM events was another measure in this experiment. When both hands *can* be used at the same time, the amount that they *are* used at the same time can be a good indicator of cognitive load. Parallelism was measured by calculating the percentage of time that both mice were active simultaneously. To be considered active, a mouse event had to occur within two milliseconds of the previous event from the same mouse. For the two mice to be considered simultaneously active, the two mice both had to be active during the same time interval.

Table 4.10 shows the average percentage of mouse events per trial that occurred during parallel action under the two dual-mouse conditions. The breakdown of events into parallel, left-mouse continuous, right-mouse continuous and non-continuous is displayed in Figure 4.19. These results support H3: participants engaged in more parallel activity under the SYM condition than under the ASYM condition. The RM ANOVA for percentage of parallel activity gives an F-ratio for condition of 1164.0 ($p_{1,954} \leq 0.0001$), showing that these results are

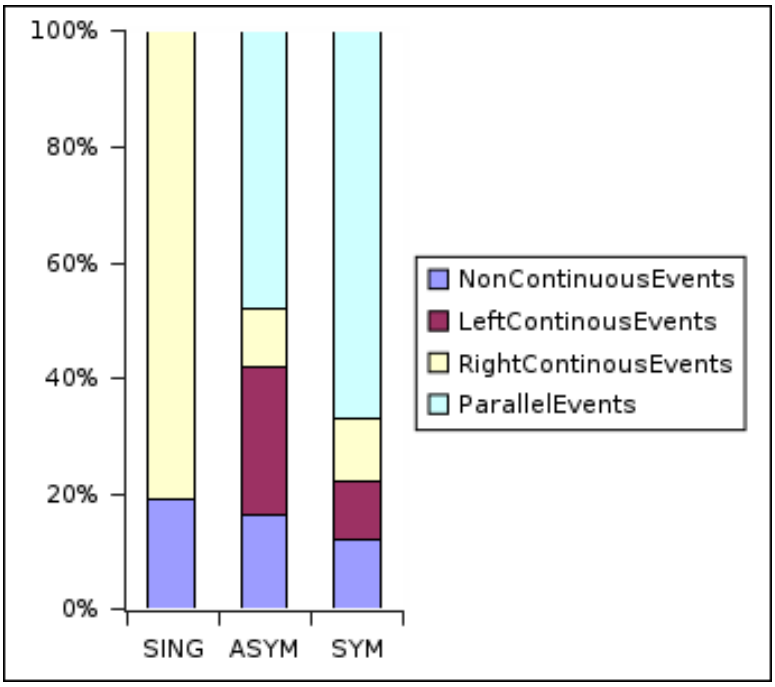


Figure 4.19: Image registration experiment: Percentage of events considered parallel, left mouse continuous, right mouse continuous and non-continuous for each condition.

significant. This also supports the conjecture that the ASYM condition imposes a higher cognitive load on the participant than the SYM condition.

Learning Effects

The 24 participants were split into six groups and the presentation order of the three conditions across the groups was counterbalanced. Table 4.11 shows the mean trial time for each condition presentation order. There is little variance across the six groups, and this is confirmed by the repeated measures analysis for total trial time, which gives an F-ratio for presentation order

Condition	% Parallelism	Std Dev
ASYM	48.2%	15.7%
SYM	67.1%	8.6%

Table 4.10: Image registration experiment: Percentage of mouse events considered parallel (occurring within two milliseconds of previous events from both mice).

Condition Order	mean total time	stdev
SYM, ASYM, SING	13.88	11.45
SYM, SING, ASYM	13.69	10.49
ASYM, SING, SYM	13.48	11.93
ASYM, SYM, SING	13.49	13.44
SING, SYM, ASYM	13.28	9.45
SING, ASYM, SYM	15.88	20.98

Table 4.11: Image registration experiment: Mean completion times (seconds) by condition order.

of 2.12 ($p_{5,1908} = 0.06$), which is not statistically significant. The final group has the most obvious deviation, with a mean total time approximately 2 seconds longer than the other groups. On closer examination, the participant who had the highest mean total trial time was in this group. That participant's mean total trial time was 22.12 seconds, whereas the overall mean total time across all participants was 13.95 seconds. Given the statistical insignificance of this difference across the groups, the learning effects are minimal.

Image Set Effects

The four images that we chose for the experiment task are shown in Figure 4.13. Each of the four images was presented to each participant 30 times during the experiment. Image alignment on the first three images was expected to be relatively easy, but participants were expected to take longer to align Image 4. I expected the completion times for Image 4 would exhibit larger differences between the conditions. As shown in Figure 4.20, the total completion times are longer for trials using Image 4, and the difference is more pronounced for the SING technique, where the 21.27 second average completion time is approximately 3 seconds longer than the completion times for the other images. The difference for both the SYM and the ASYM techniques is approximately one second. This suggests that the performance benefits from using dual-mouse techniques in this task become more pronounced with task difficulty.

Participant Evaluation

After completing the experiment, participants were asked to rank the three interaction techniques. Specifically, they were asked, "Imagine you had a job lining up digital images all day. Rank the three techniques in order of preference according to which you would like to use in your job." The participants were not told of their performance with any of the techniques before answering this question. The symmetric technique was preferred by 22 of 24 candidates. One of the 24 participants preferred the single-handed technique and one of the

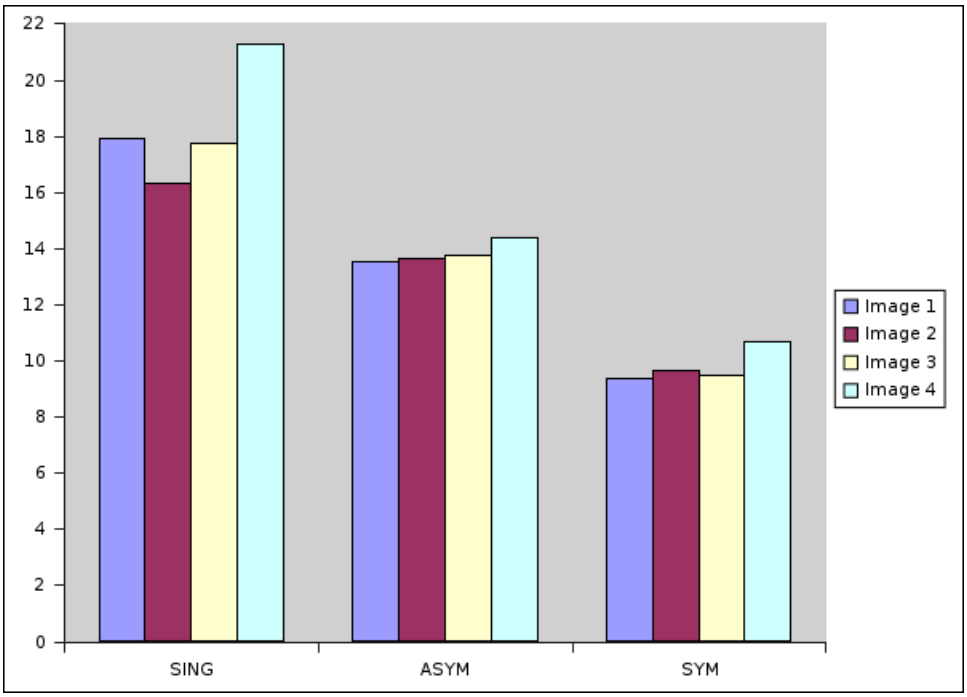


Figure 4.20: Image registration experiment: Mean completion times (seconds) by condition and image.

participants preferred the asymmetric technique (although both performed fastest with the symmetric technique). This preference ranking further supports the assertion that the task is naturally symmetric.

In addition, participants were asked if they had any comments about the experiment in general or any of the specific techniques. Most of the comments were positive regarding the dual-mouse conditions, such as "It was weird at first using two mice, but then it was easy" and "It was a lot easier with two mice" and "The symmetric technique was best". All of the comments are recorded in Appendix A.

4.5.4 Image Registration Experiment Conclusions

The results of the image registration experiment demonstrate that symmetric interaction provides better performance and higher user satisfaction for rotating, translating and scaling objects in two dimensions, compared to using a single mouse or a dual-mouse asymmetric technique.

4.6 Basic Geometry Conclusion

This chapter presented symPut, a driver to support the development of symmetric interaction techniques and applications, and symDraw, an application for drawing 2-D vector images with a completely symmetric interface. symDraw is unique in that it requires the use of two mice to create basic shapes, which allows for faster shape creation and manipulation. It also affords users the ability to develop skill in their non-dominant hand while they are creating their images. The symDraw application also provides an prototype example of a completely symmetric, dual-cursor interface.

The rotation and translation experiments and the image registration experiment both show that symmetric interaction provides an efficient method for controlling 2-D objects on screen. Although these techniques are not new and were briefly described by Kurtenbach et al [KFBB97], they had never before been formally evaluated. Participants in both studies performed the object manipulation tasks more efficiently using the symmetric techniques. While the subjective results from the rotation and translation experiments were inconclusive, participants in the image registration experiment largely preferred the symmetric interaction technique. These results support the hypothesis of the thesis: that users will perform symmetric tasks faster with symmetric interaction techniques and will prefer the symmetric interaction techniques for such tasks. In the next two chapters, I present new symmetric techniques that were created based on the Symmetric Interaction Model.

Chapter 5

Spline Manipulation

In the previous chapter, I presented symDraw, a symmetric interface for 2-D drawing. The shape creation and manipulation techniques that are the basis of symDraw were presented, as well as experimental validation of the effectiveness of those techniques. The symSpline technique that is presented in this chapter was added to symDraw in the later stages of development. The symSpline technique allows a user to interactively edit a spline using two mice and two cursors.

The first two sections of this chapter include a brief introduction to splines and the previous work in spline manipulation techniques. Following that is a thorough description of the symSpline technique. Then a formal experiment designed to test symSpline against single-mouse and dual-mouse asymmetric techniques is described. The chapter ends with a discussion of the experiment results and the contribution of these results to the problem posed in the introduction to this thesis.

5.1 Introduction to Splines

A spline is a curve that is defined by a mathematical function, usually a sum of polynomial basis functions. The purpose of using splines is to have a curve with certain properties, such as higher order continuity. These properties are often desirable in industrial design settings, where smooth curves are important for both scientific (aerodynamics, flow, etc.) and aesthetic reasons. Curve creation tools are also part of standard graphics programs such as Adobe Illustrator, PaintShop Pro and the Gimp. Because of this, novice users often try to use the splines tool when they want to create a curve in their drawings. However, creating a desired spline curve is a difficult task that requires considerable skill. I believe that the spline interfaces in most traditional graphics packages are difficult for novices to learn. There are a number

of problems with these interfaces that make curve drawing and editing challenging for novice users and inefficient for expert users. These problems include:

Choosing a manipulation method Splines can be manipulated in a variety of ways in traditional graphics programs: by moving control points, by moving edit points, by adjusting the slope of a tangent to an edit point, or by adjusting the length of a tangent to an edit point. Deciding which manipulation to perform first is difficult: should the edit point be translated, and then the slope of the tangent adjusted? Or should it be done in the opposite order? This thought process is time consuming and difficult for the novice user.

Constant mode-switching The second problem with traditional one-handed spline manipulation is that it is time-multiplexed. Users can only perform one type of spline manipulation at a time and some type of mode switch is required to change the editing tool. The mode switch is both distracting and time-consuming, whether achieved by pressing keyboard modifier keys or by clicking on a toolbar button or selecting a menu option.

Non-intuitive interaction The third problem is that the result of the chosen manipulation is often far from what is expected by the novice user. This is especially true for indirect manipulation methods where the user moves control points that are not on the curve. However, even the direct manipulation of splines (via editing points on the spline rather than control points) can be difficult for novice users who are not familiar with tangents and curves.

The symSpline technique is designed to address these issues by getting rid of mode-switching and providing a single manipulation method so that users do not have to choose *how* to manipulate the spline. These two factors increase the ease of exploration of the spline curves and the intuitiveness of the interface. Because the user has the opportunity to explore many spline configurations easily, he may learn what a given manipulation will do to the spline curve more quickly with the symSpline technique than with traditional techniques.

5.2 Previous Work in Bimanual Spline Manipulation

Direct manipulation of splines is not new. The technique was first introduced in 1989 by Bartels and Beatty [BB89]. The technique was refined and tested to include the ability to adjust the frequency of a spline segment by increasing the length of the tangent at the selected point [FB93, JBFG93]. Since that time, numerous researchers have used the method and applied it to 3-D surfaces and to specific modelling problems. Finkelstein and Salesin [FS94] made important contributions by addressing issues of locality of control and preservation of detail, while allowing direct manipulation of splines. These techniques were asymmetric and

made use of a single mouse, sometimes with the addition of keyboard controls. Fowler made use of a dataglove for direct manipulation of splines, which allowed 3-D control of tensor product surfaces [Fow92]. Although this was an important contribution, the manipulation was asymmetric and required a specialized input device.

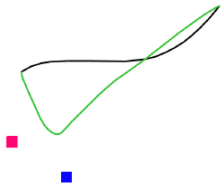


Figure 5.1: An example from the spline manipulation of Owen et al. The participants had to manipulate the spline by moving the pink and blue control vertices. Included with permission of Russell Owen.

Owen et al. used spline manipulation as a test case for examining the differences in performance between bimanual and unimanual interaction techniques [OKF+05]. They were also interested in the difference between integrated and separate workspaces for the two hands in bimanual interaction. Their experiment did not involve direct manipulation of splines; participants were asked to match splines by moving control vertices shown as pink and blue squares in Figure 5.1. In addition, they used asymmetric input devices: a stylus in the dominant hand and a puck in the non-dominant hand. They found no difference in performance related to whether the devices used separate workspaces (each using a specified half of a drawing tablet), or an integrated workspace (sharing the whole drawing tablet). They found the two-handed techniques to be faster than the unimanual technique. However, when switching times for the single-device techniques were removed, the single-device technique was faster than the two-handed techniques. This implies that there are no cognitive benefits to the two-handed approach to the task, only movement time savings. This is counter to what the authors expected. The two-handed interaction technique was symmetric in that the task for each hand was to drag a control point around. However, the spline was not visually integrated, and so the user would consider each hand to be performing a separate task of moving a control point. I believe that the lack of visual integration, combined with the use of asymmetric devices, explains why there were no cognitive benefits associated with the two-handed techniques tested.

5.2.1 Specialized Device Spline Manipulation

Some studies that focus on direct manipulation of splines using specialized devices have been done. ShapeTape, a specialized high degree-of-freedom curve input device, has been used for creating curves and surfaces as well as for defining specialized gestural input for interface

control [BFKS99, GBS03]. Singh developed a digital French curve tool manipulated on a digital tablet with a stylus and puck [Sin99]. This technique mimicked the curve templates used by designers in the real world. The ShapeTape tool allowed both symmetric and asymmetric interaction, while the digital French curves allowed only asymmetric interaction. While both of these interaction techniques provide interesting and efficient methods for interacting with spline curves, neither of these specialized devices are likely to be accessible on the typical computer desktop.

Recently, there have been two applications of two-handed constraint-based direct manipulation of splines, both targeted at the automotive design field. The first was the work on digital tape drawing by Balakrishnan et al. [BFKB99, GBK+02]. In real tape drawing, the user's two hands asymmetrically apply electrical tape to a large wall display to create curves. To edit a curve, the tape is unstuck and reapplied at a different angle. The digital tape drawing system used two trackers and a large digital wall display. The method was not designed to follow Guiard's guidelines for asymmetric interaction; instead, it mimicked the way traditional tape artists worked with real tape. The tape artists (whether left-handed or right-handed) used their right hand to set the frame of reference in which the left hand worked, and the left hand worked at a higher spatial frequency than the right hand. The precedence of action started with the left hand, but later switched to the right hand. Thus, although the interaction was asymmetric in both the traditional and digital tape drawing, it did not follow the Guiard model of asymmetric interaction. However, tape art is a very specialized skill and it is likely that it is taught only one way, regardless of the handedness of the artist. This work showed the benefit of using both hands for the creation of digital splines as well as the importance of carefully studying the real-life interaction technique before implementing a digital version.

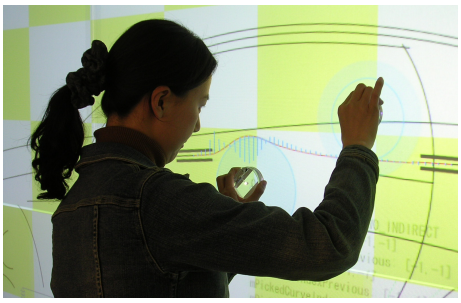


Figure 5.2: Bae et al.'s wall-size spline manipulation technique with specialized devices. Image included with permission of Bae.

Inspired by the digital tape drawing interface, Bae et al. used two graspable handles to do fine-tuning of spline curves drawn on a large wall display [BKKK04]. Their interaction technique was asymmetric and followed Guiard's guidelines. They used specialized circular devices, as shown in Figure 5.2, and a language of gestures to specify spline manipulation

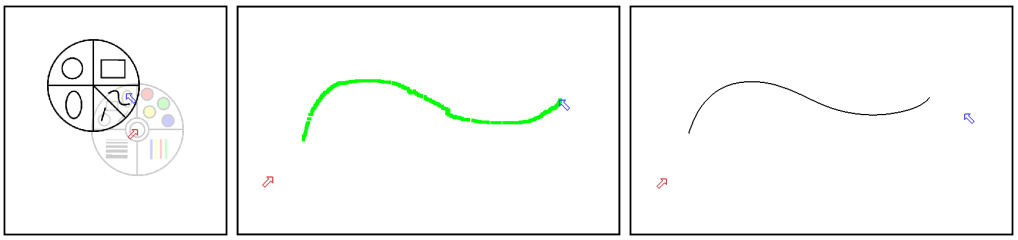


Figure 5.3: In symDraw, users create a spline by sketching a path with one mouse.

operations.

5.2.2 Previous Work Summary

The initial work done in splines set the stage for interactive spline manipulation. However, many different methods for manipulating splines were developed and the best were incorporated into traditional graphics packages. This has meant that users have to choose how to manipulate splines and have to spend time switching between these different tools. Some specialized devices have been used in attempts to simplify the manipulation of splines, or to make the manipulation more expressive. However, these techniques have not been integrated into the standard desktop, likely because of the expense and the need for specialized equipment.

The two large-scale wall techniques have demonstrated the benefits of using two hands for spline manipulation, but have an audience limited to automotive design engineers. The symSpline technique aims to scale two-handed spline manipulation down to the desktop, with a regular display and with regular mice, rather than specialized devices. Two-handed spline manipulation could then be used by a more general audience including artists, animators and everyday computer users. Finally, my method differs from the one used by Owen et al. in that I allow direct manipulation of the spline curves and provide full visual integration between the cursors controlled by each hand.

5.3 The symSpline Technique

The symSpline technique is a symmetric interaction technique that allows users to explore and adjust curves in a fast and intuitive manner. With symSpline users don't have to decide which tool to use for spline manipulation because one tool is all that is required. This means users don't have to spend time switching between tools. In this section, I will describe how users create splines in symDraw and how they edit the spline once it has been created.

5.3.1 Spline Creation in symDraw

To create a spline in symDraw, the user chooses the spline tool from the shapes sub-menu. Then the user sketches out a spline using one mouse, either their left or right, depending on hand preference. Because left-handed users can sketch splines with their left mouse and right-handed users can sketch splines with their right mouse, this interface addresses the goal of being free of handedness bias. As they hold down the main button and drag the mouse, green dots appear, marking the path that they have created. When they release the mouse button, the path of dots is automatically turned into a spline curve. This transformation is achieved by averaging every ten green points to create a B-spline control point. However, there are special cases at the ends: the first green point becomes the first B-spline control point and the last green point becomes the last B-spline control point. A uniform knot vector with full multiplicity at the ends is used with the control points to create an end-interpolating spline. The spline creation interface is shown in Figure 5.3.

5.3.2 Spline Editing

The curves that are created in symDraw are endpoint-interpolating cubic B-splines. The splines can only be manipulated directly, not by moving control points. The algorithm used for the constraint-based manipulation of the spline curves comes from the thesis work of Fowler [Fow90]. Fowler's algorithm takes a change in a point on the curve or the tangent to the curve and makes the minimum adjustment to the control points to achieve this change.

Once a spline has been created, the user can select a point on the spline by clicking on the spline with the middle button on either mouse. At this point, a tangent to the point appears and the user's two cursors warp to the ends of the tangent. Note that while the spline is being edited, the spline's control points are visible as green dots on the screen. This is useful for expert users and may help novice users learn to understand the spline's structure. The user can now move the two cursors around to translate, scale and rotate the tangent and the spline adjusts interactively to fit the tangent. Figure 5.4 shows an example of this process where the user is being very exploratory and even ties the spline in a knot while exploring what that segment of the spline can do.

5.3.3 symSpline Interaction Details

To manipulate a spline, the user starts by selecting a point P on a curve. After selecting P, the ends of the tangent to P are locked to the two cursors and P is locked to the midpoint of the tangent. By moving the tangent around with two mice, three separate effects are available:

- Point translation – as P moves, the spline segment moves.

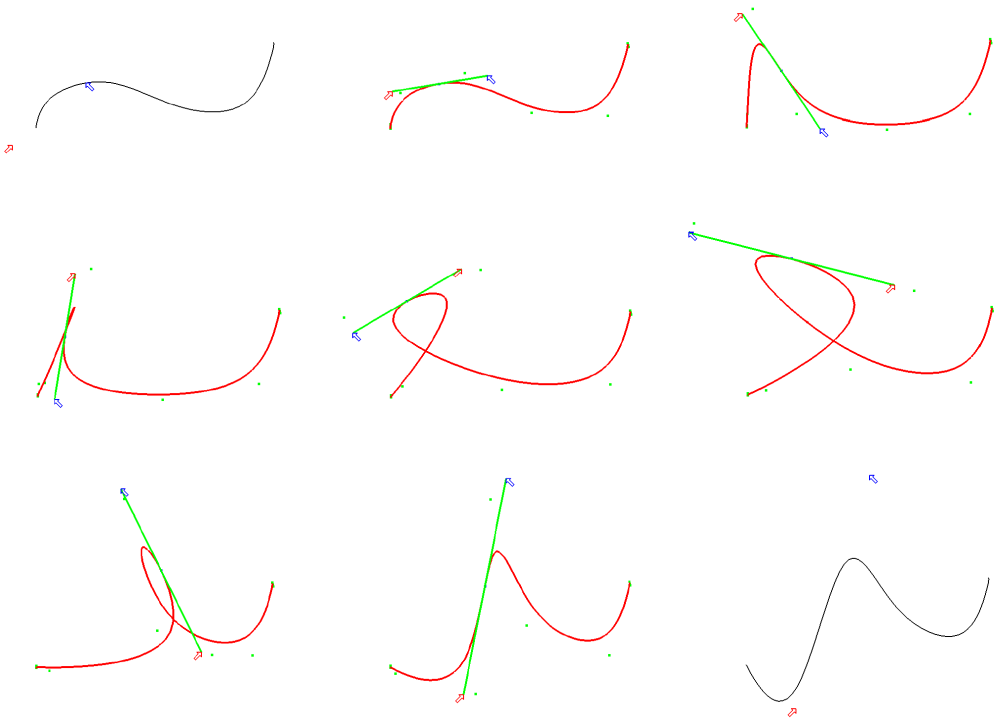


Figure 5.4: After selecting a point to edit, the user's two cursors control the ends of a tangent to that point. The spline updates interactively as the tangent is manipulated.

- Tangent rotation – as the angle of the tangent changes, the spline adjusts such that the first derivative at P matches the slope of the tangent.
- Tangent scale – as the tangent lengthens, the spline segment gets wider; as the tangent shortens, the spline segment becomes more narrow.

Figure 5.5 shows the first two effects in serial: a spline that is relatively straight is made into a curve by moving the edit point up, and then it is skewed by adjusting the angle of the tangent. Figure 5.6 shows the last effect. A curve of medium width can be widened by pulling the tangent ends apart or narrowed by pressing the tangent ends together.

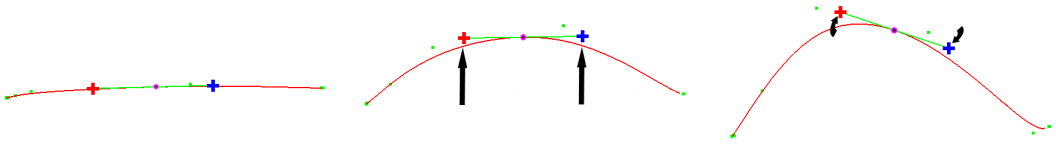


Figure 5.5: A straight curve is adjusted by moving the edit point up and then rotating the tangent to skew the curve.

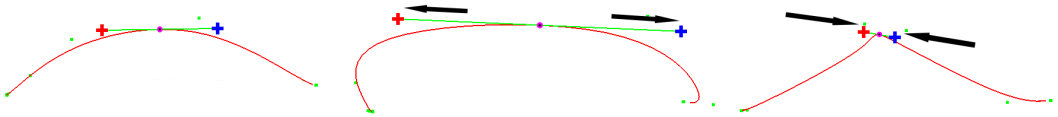


Figure 5.6: A medium curve is made wide or narrow by stretching or shrinking the tangent.

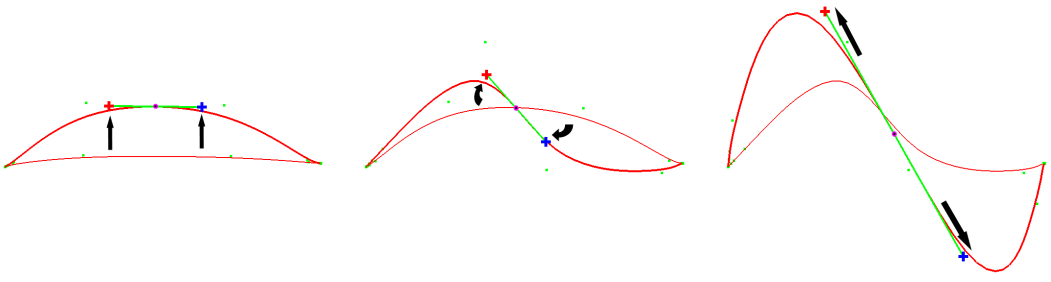


Figure 5.7: Three tangent movement effects applied on a straight curve to twist it into an 'S' shape. Previous curve state is shown at each stage for illustration purposes.

Because symSpline uses cubic B-splines, the effects above can be applied to the middle of a straight curve to twist it into an 'S-curve' in a symmetric motion (Figure 5.7). The nature

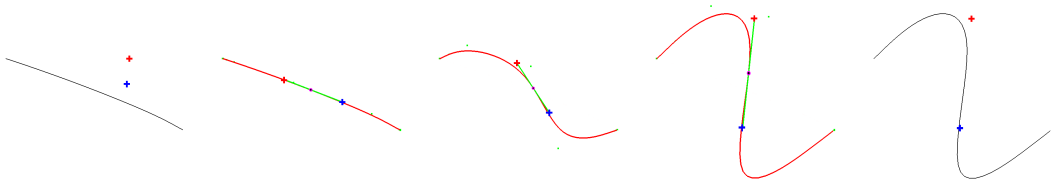


Figure 5.8: The tangent movement effects applied simultaneously.

of the task and the symmetric design of symSpline allow all three effects to be combined in a single two-handed symmetric motion. The three effects do not have to be serialized; indeed, it is expected that expert symSpline users will combine the effects simultaneously as illustrated in Figure 5.8.

5.3.4 Advantages of symSpline

The symmetry of the symSpline technique allows a user to perform point translation, tangent scaling and tangent rotation all in a single, symmetric gesture (Figure 5.8). This means that novice users need not spend time deciding what type of manipulation to perform (addressing the first problem in Section 5.1). Both novice and expert users can avoid the cost of switching modes (addressing the second problem in Section 5.1). Thus, the symSpline technique affords easy exploration of the effects of the different types of manipulation. Because exploration is easy, it is likely that users will be able to more quickly understand the effects of different manipulations (addressing the third problem in Section 5.1).

The symSpline technique affords curve scaling in a manner that mimics the way people use their hands symmetrically to stretch objects in real life. Additionally, the tangent in symSpline is pushed around the screen by the two cursors similar to the way a bike is steered by the two hands holding the ends of the handlebar. This steering metaphor is effective because it mimics the way people naturally move large objects in space. The symSpline technique fits nicely into the symDraw application because the tangent is manipulated in exactly the same way as basic lines are manipulated (Figure 4.2), allowing learning transfer across the techniques.

The advantage of two-handed techniques over traditional single-mouse techniques is the ability to explore many different curves. Once a point has been selected, the two-handed techniques allow a user to move the point freely while simultaneously exploring the tangent adjustments, without requiring a mode switch. Owen et al. described this exploration ability as the 'expressiveness' available in two-handed input [OKF+05]. In contrast, the single-mouse approach requires a mode switch between the translation of the point and the adjustment of the tangent, which makes the process of exploring the possible point locations and tangent adjustments tedious, and is likely to result in fewer possibilities being explored and a less satisfactory final curve.

The benefit of the symSpline technique over the dual-mouse asymmetric techniques that were used in the experiment described in Section 5.4 is that the user does not have to remember which hand is performing which task. The tangent stretching between the two hands visually integrates the tasks of each hand into a single, perceptually unified task. The symmetry allows the user to concentrate on moving the tangent around in space to achieve the desired curve, rather than trying to remember to translate the point with one hand and adjust the tangent with the other.

5.3.5 Dual Cursor Issues

There are some issues that can arise with dual-mouse, dual-cursor interaction techniques such as symSpline. The first issue is divided attention: if the user has to keep track of two separate cursors on screen, she must decide which cursor to pay attention to at any given time, and may end up serially switching attention back and forth between the two. This issue was studied by Balakrishnan and Hinckley, who recommended that the objects controlled by the two hands be visually integrated [BH00]. Because the symSpline technique provides visual integration between the two cursors through the tangent (at least during spline manipulation), divided attention issues do not arise.

The second dual cursor issue is the relative position of the cursors on screen, with respect to the user's hands. If the cursor controlled by the left hand moves to the right of the cursor controlled by the right hand, the user can become confused because people generally expect things controlled by their right hand to be to the right of things controlled by their left hand. This problem was discussed in Chapter 4 in relation to the simultaneous rotation and translation experiment. In this experiment the stimulus-response incompatibility is most likely to occur when the splines are oriented vertically. However, there are a number of factors acting to mitigate this issue. First, because the dual-mouse spline manipulation techniques provide visual integration between the two cursors in the form of the tangent, it seemed likely that the users would be able to handle the stimulus-response incompatibility without large degradation in performance. The users would learn to handle the reverse mapping by considering the tangent to be 'flipped over'. The second factor is that the stimulus (moving the mouse) and the response (the tangent moving on screen) are both continuous – stimulus-response compatibility issues are much more likely to occur when the stimuli and responses are discrete. However, a small degradation in performance seemed possible, as this was seen in the rotation and translation experiment reported in Section 4.4.

5.4 symSpline Experimental Evaluation

I designed an experiment to test the symSpline technique against other spline manipulation techniques. I chose a simple spline-matching task so that novice users would be able to participate (expert spline users are few in number, and difficult to find). As part of this thesis, it was necessary to test symSpline against both the traditional one-handed technique and two-handed asymmetric techniques.

5.4.1 Techniques Tested

The methods tested in the experiment are summarized in Table 5.1. The experimental task involved matching two spline curves, similar to the task recommended by Bartels et al. for splines experimentation [BBB+93]. Matching was performed by translating a predetermined edit point on a movable spline and adjusting the tangent until the movable spline matched a fixed target spline. One difference between Bartel's spline matching task and the task in this experiment is that the two splines share the same screen space rather than being in two separate side-by-side windows. This would allow closer matching, and prevent fatigue from switching focus between two different screens. This experimental setup is similar to the one used by Owen et al. [OKF+05]. The one-handed technique chosen was based on the technique used in current graphics software including Adobe Illustrator, where the cursor can be used either to move the edit point or to adjust the tangent. In the asymmetric-left technique, the left mouse controlled point translation and the right mouse controlled tangent adjustment. The asymmetric-right technique reversed the task assignment.

Technique	Point Translation	Tangent Adjustment
Single Mouse	Select point and drag with main button pressed	Select end of tangent and drag with main button pressed
Asymmetric Left	Drag left mouse with main button pressed	Drag right mouse with main button pressed
Asymmetric Right	Drag right mouse with main button pressed	Drag left mouse with main button pressed
symSpline	Unified manipulation: drag both mice with main buttons pressed	

Table 5.1: Techniques tested in experiment. 'Main' button refers to the left button on the right mouse and the right button on the left mouse.

Asymmetric techniques

Two asymmetric techniques were included in the experiment because it was unclear in the pilot testing stage which of the two asymmetric techniques was more natural. Of the two sub-tasks (point translation and tangent adjustment), the tangent adjustment sub-task seems to require more fine-grained control. This indicates that the asymmetric-left technique, where the left hand controls the translation of the point and the right hand controls the adjustment of the tangent would be most natural for our right-handed participants. Additionally, since pilot participants appeared to move the point first and adjust the tangent second when using the asymmetric techniques, having the left hand do the task that comes first fits with Guiard's guidelines. However, most pilot participants complained that this technique was difficult to work with and that they preferred the opposite technique, where the right hand translates the point and the left hand adjusts the tangent. The completion times for these pilot test participants were also better for the asymmetric-right than for the asymmetric-left. Because there was no clear winner between the two dual-mouse asymmetric techniques, both were included in the final experiment. The fact that there is no obvious way to divide the task asymmetrically suggests that the task is naturally symmetric. This supports the rule of thumb presented in the Symmetric Interaction Model (Section 3.3): if it is difficult to determine appropriate task assignments for the dominant and non-dominant hands, the task is probably naturally symmetric.

5.4.2 The Spline-Matching Task

Because splines are relatively specialized, and the test participants were not familiar with how to manipulate them, it was necessary to make the spline matching task as simple as possible, while still testing the effectiveness of the interaction techniques. Specifically, it was important to ensure that the timing results would not be affected by the level of splines knowledge and experience. To prevent this type of variance, the participants were limited to manipulating a single point at the center of a movable spline. In this way, the participants did not have to make a value judgment about what part of the spline to select for manipulation.

There were two other simplifications to the spline matching task. First, the end points of the two splines were clamped together and the participant could not adjust the end points of the movable spline. Second, the splines used for the experiment were simple. All of the target splines were single-segment cubic B-splines. The moveable splines were dual-segment cubic B-splines, because single-segment splines proved difficult for pilot participants to match. These factors combined to make the task easy enough that participants with no prior splines experience could quickly master the task.

The participant's task was to acquire the highlighted edit point in the middle of the blue spline (Figure 5.11(c)) and then manipulate the blue spline until it matched (*i.e.*, was on top

of) the green target spline (Figure 5.11(d)). A spline match was obtained when the maximum of the Euclidean distances from each point on the movable spline to the nearest point on the target spline was under ten pixels. Thus the match was geometric and not parametric. Note that a match could be obtained if the edit point was placed within a range near the middle of the target spline.

The edit point was always acquired by clicking on it with the main mouse button (left button on right mouse or right button on left mouse). Once the point was acquired, the tangent appeared, and the cursor(s) warped appropriately. All manipulations required the user to hold down the main mouse button(s) and drag to move the tangent and/or edit point. In the single-mouse technique, the user could either click down on the edit point and drag to translate, or click down on the highlighted end of the tangent and drag to adjust the angle and length of the tangent.

5.4.3 Test Splines

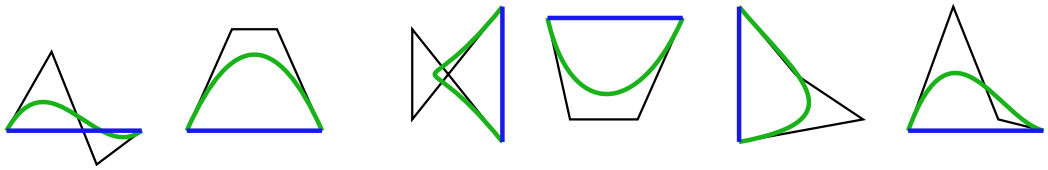


Figure 5.9: Practice splines. The control polygon of each spline is shown here, but was not displayed to participants.

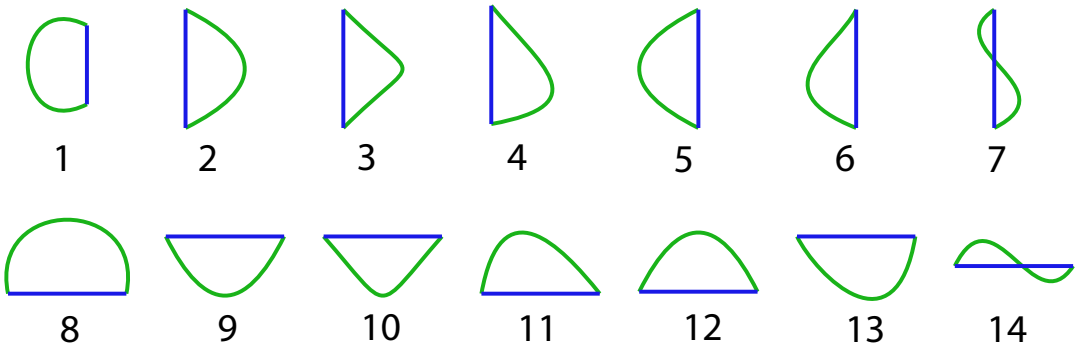


Figure 5.10: Splines used in timed trials.

The fourteen splines shown in Figure 5.10 were used as the input data for experiment. The six splines shown in Figure 5.9 served as inputs for the practice trials. The test splines have four main characteristics, being either wide, medium or narrow in the shape of the curve, being either centered or skewed in the alignment of the curve, and having either one or two bumps (hereafter a spline with two bumps will be referred to as an S-curve). This variety was desirable for two reasons. First, using a variety of different splines would help to prevent motor learning effects from the participants repeating the same motions for the same splines. Second, skewed splines and S-curve splines would likely be harder to match and would therefore reveal how the techniques differ when the task is more challenging. More complex splines were considered, but the addition of more control points during pilot testing changed the nature of the task and was too difficult and time-consuming for novice participants. In addition, it was desirable to limit the splines to those that could be matched (from a straight line) with a single motion.

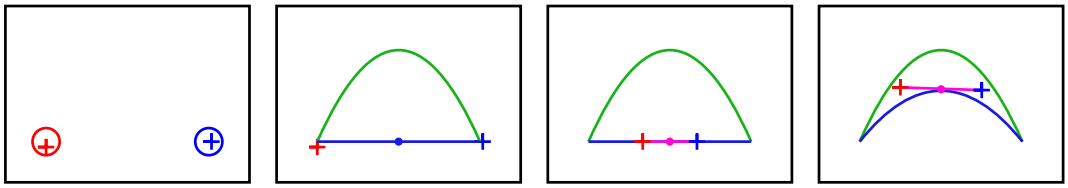
5.4.4 Trial Setup

The participant pool consisted of thirty-four undergraduate students from a variety of departments (Arts, Accounting, Health, Science, Biochemistry, Mathematics) at the University of Waterloo. Nineteen of the students were female, sixteen were male. All participants self-identified as right-handed.

Because the participants were not expected to have any familiarity with manipulating spline curves, they were given some instruction before the experiment. They were shown an online presentation that contained examples of spline curves and how they could be changed by moving an edit point, adjusting the angle of a tangent and adjusting the length of a tangent. The presentation also gave the participants general information on how the experiment would proceed.

The experiment session was a within-participants design, consisting of four blocks of trials, one for each technique described above. In the first block there were six practice trials and in each of the subsequent three blocks, there were three practice trials (Figure 5.9 for pictures of the practice splines). The increased practice trials in the first block were used to help the participants become accustomed to the splines. Each set of practice trials was followed by 28 randomized trials, consisting of 14 different splines, each seen twice (Figure 5.10). Thus, each participant performed a total of 15 practice matches and $14 \times 2 \times 4 = 112$ timed matches during the experiment. The order of the four blocks was randomized for each participant.

Each experiment session lasted roughly 40 minutes, including the time to sign consent forms (Appendix B), view the online instructions and ask questions. Most participants found the instructions sufficient and did not ask any questions. A few participants took as little as 30 minutes to complete the experiment and one participant took almost 55 minutes to complete the experiment. The participants were paid $10 for their participation.



(a) (b) (c) (d)

Figure 5.11: Trial procedure for symSpline.

Figure 5.11 shows the process of a trial. At the start of each trial, one or two small hollow circles appeared, and the participant had to move their cursor(s) into the circle(s) (Figure 5.11(a)). When the cursor(s) were inside the circle(s), four evenly-spaced beeps were played and the trial began at the end of the fourth beep. At this point, the blue movable spline and the green target spline both appeared on screen (Figure 5.11(b)). The two splines had their endpoints clamped together. The blue spline started off as a straight line stretching between the two endpoints of the green target spline. The spline endpoints were located where the start circles had been located, thus the participant's cursors always started at the ends of the splines.

5.4.5 Hypotheses

The following results were predicted:

- H1: The symSpline technique would be faster than the other three techniques.
- H2: The dual-mouse asymmetric techniques would be faster than the single-mouse technique, but slower than the symSpline technique.
- H3: The symSpline technique would be the most preferred technique in the participants' post-experiment evaluation.

5.5 Results and discussion

The thirty-four participants completed 112 timed trials each, for a total of 3808 trials. All three hypotheses are strongly supported by the results of the experiment.

5.5.1 Outlier Removal

Out of the 3808 trials, 134 trials (3.2 percent) were removed as outliers, being greater than three standard deviations from the mean trial completion time. These outliers were not correlated with any of the four conditions or with any of the participants or spline IDs. Thus, the remainder of the analysis was performed using 3674 trials.

5.5.2 Condition Presentation and Learning Effects

The within-participants design made it necessary to check that there were no confounding effects due to the order of presentation of the techniques. After aggregating the results for each participant for each of the four conditions, an ANOVA of average trial completion time was performed, using condition and block number as variables. The condition variable was significant ($F_{3,120} = 12.1, p \leq 0.0001$). The block number variable was not significant ($F_{3,135} = 0.59, p = 0.62$), showing that there were no significant learning effects. In addition, there was no interaction effect for $condition \times blocknumber (F_{9,120} = 0.51, p = 0.8673)$. This shows that the random assignment of condition presentation order was sufficient to prevent order effects, and so the block number variable could be removed from the analysis.

5.5.3 General Results

Source	DF	F Ratio	Prob > F
Technique	3	98.46	< .0001
Alignment	1	111.27	< .0001
ID	6	46.85	< .0001
Alignment × Technique	3	8.86	< .0001
ID × Technique	18	13.90	< .0001
Alignment × ID	6	16.01	< .0001
Technique × Alignment × ID	18	1.04	.4059

Table 5.2: Repeated Measures Analysis of Variance for completion time against technique, spline ID and alignment, with interactions. DF Denominator is 1806 for all variables (1895 data points included).

After removing outliers and ensuring that learning and condition order were not significant factors, the aggregate data were analyzed. These data points consisted of the four conditions for each of the 14 splines for each participant (effectively splitting the original data set in half, since each participant saw each spline twice for each of the four techniques). The hypothesis

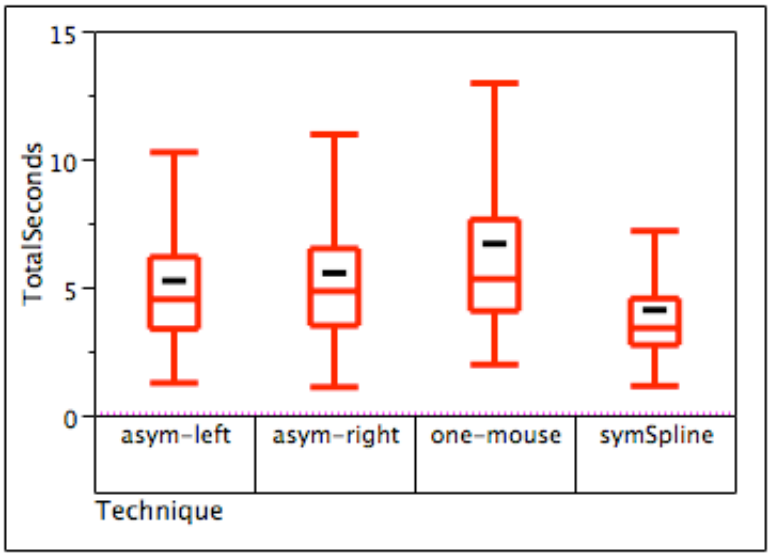


Figure 5.12: Boxplots of trial completion time for the four different techniques. The mean trial completion time is shown by the black dash inside of each box.

that the symSpline method would be faster than the other three techniques is supported by the aggregate participant results as summarized in Figure 5.12 and the repeated measures ANOVA shown in Table 5.2. Tukey HSD pairwise tests also indicate that the symSpline technique is significantly different from the other techniques. The mean completion time of the symSpline technique is more than 2.5 seconds faster than the mean completion time of the single-mouse technique.

The second hypothesis, that the dual-mouse asymmetric techniques would be faster than the single-mouse technique, is also supported. The two asymmetric techniques generated faster completion times than the single-mouse technique, but were not as fast as the symSpline technique. However, the difference between the two asymmetric techniques appears to be minimal and is not statistically significant in Tukey HSD pair-wise tests, which supports the argument that the task is naturally symmetric.

5.5.4 Variable Analysis

To analyze the effects of the different splines and the orientation of the splines, I aggregated the data for each participant by $splineID \times splinealignment \times technique$ and analyzed this data with a repeated measures ANOVA, shown in Table 5.2. Tukey HSD Post Hoc tests confirm that the four techniques are statistically different from one another at the $\alpha = .05$ level, except for the two asymmetric techniques, which are not significantly different. These tests, combined

with the low completion time mean, show that symSpline is significantly faster than the other techniques in our experiment.

Spline Alignment Effects

The spline ID variable is a unique identifier assigned to each of the seven horizontal splines and to each of the seven vertical splines in the data set, see Figure 5.10. It makes sense for the ID to have a significant effect, as the different splines in the dataset would have different difficulty levels, depending on various aspects of the spline's shape. In addition, there is an interaction between splineID and technique. This makes sense, as it was expected that the difference between the symSpline technique and the other techniques would be more pronounced for splines that are more difficult to match. The validity of this explanation is examined in the following section. The alignment characteristic was examined to see if the vertically-aligned splines would be more challenging, especially for the dual-cursor techniques. There is a main effect of alignment and there are also $alignment \times splineID$ and $alignment \times technique$ interaction effects.

5.5.5 Spline-dependent results

Using the fourteen splines displayed in Figure 5.10 enabled analysis of completion times with respect to various aspects of the splines themselves. Two different variables are considered, in combination with spline alignment: the number of bumps on the spline (1 or 2) and the shape of the spline (symmetric or skewed). I expected the S-curve splines (shown in Figure 5.13) and skewed splines (shown in Figure 5.15) to be more challenging to match and I expected the symmetric technique to perform well with more difficult spline-matching tasks.

Single-bump versus S-curve Splines

To examine the differences between single-bump splines and S-curve splines, I selected two single-bump splines (one vertical and one horizontal) to compare with the vertical and horizontal S-curve splines. Splines labeled 4 and 11 were chosen to compare with the splines labeled 7 and 14 (Figure 5.10) because similar tangents would be required to match these splines.

When the aggregated trial completion times are separated by splines with one bump versus S-curves, salient differences appear (Figure 5.14). The difference between the symSpline technique and the single-mouse technique is especially large for S-curve splines. The symSpline technique is actually faster with S-curve splines than it is with single-bump splines, which is not the case for the two asymmetric techniques. It is unclear why the symSpline technique

Figure 5.13: A single-bump spline and an S-curve spline.

performs better with the s-curve than with the single-bump spline. To analyze the significance of these differences, a repeated measures ANOVA was done using trial data for the four splines aggregated over the 34 participants. The results of this analysis are shown in Table 5.3. There is an interaction effect of $Bumps \times Technique$, showing that the difference between single-bump and S-curve splines for the four techniques is significant. A Tukey HSD Post Hoc test shows that the symSpline technique is significantly different from the other techniques for S-curve splines at the $\alpha = .05$ level.

Source	DF	F Ratio	Prob > F
Technique	3	89.85	< .0001
Alignment	1	.83	.3626
Bumps	1	76.30	< .0001
Alignment × Technique	3	7.80	< .0001
Bumps × Technique	3	44.37	< .0001
Alignment × Bumps	1	2.05	.1521
Technique × Alignment × Bumps	3	0.44	.7210

Table 5.3: Repeated Measures Analysis of Variance for completion time against technique, alignment, and bumps with interactions. DF Denominator is 492 for all variables (541 data points included).

Skewed versus Symmetric Splines

To examine the differences between skewed splines and symmetric splines, splines labeled 2, 5, 9 and 12 were selected to represent symmetric splines and splines labeled 4, 6, 11 and 13 represented skewed splines (Figure 5.10).

When the mean trial completion time results are separated by symmetric splines versus skewed splines, there are no significant differences across the four techniques (Figure 5.16). Although skewed splines take longer to match than symmetric splines, this difference does not

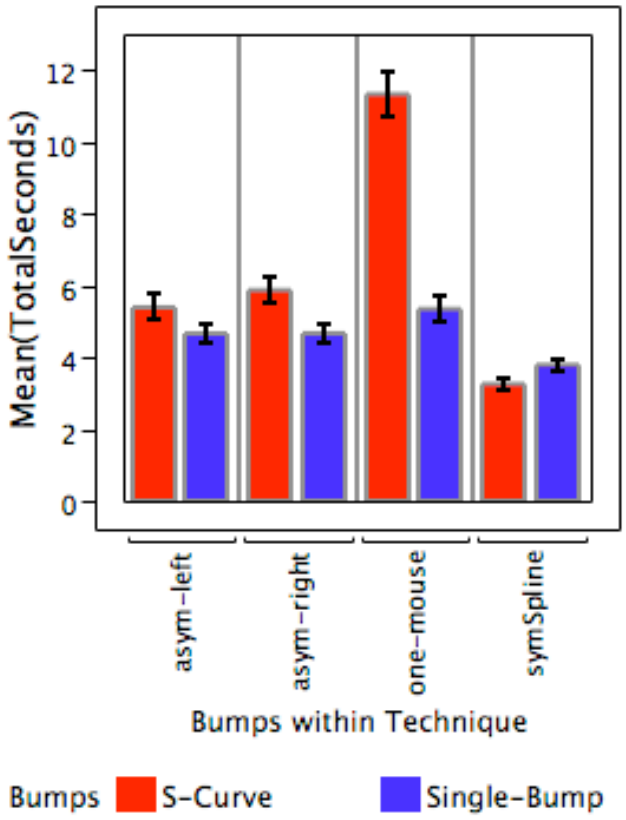


Figure 5.14: Mean trial completion times for single-bump versus S-curve splines for each of the four techniques.

vary with the technique being used. Thus, our conjecture that the harder to match splines would benefit most from the symSpline technique is not supported by this data. A repeated measures ANOVA for the four spline sets in this analysis has an $F_{3,1034} = 0.37, p = 0.7753$ for $technique \times skew$, showing that there is no significant difference for this interaction (Table 5.4).

5.5.6 Subjective Results

The third hypothesis, that the symSpline method would be preferred by participants, is also supported. After completing the experiment, users were asked to rate the four methods in terms of usage preference (Table 5.5). Specifically, the participants were asked:

> If you had a job where you had to manipulate curves like those in the experiment, which method would you most like to use, and least like to use?

Figure 5.15: A symmetric spline and a skewed spline.

Source	DF	F Ratio	Prob > F
Technique	3	18.12	< .0001
Alignment	1	96.38	< .0001
Skew	1	52.18	< .0001
Alignment × Technique	3	5.56	.0009
Skew × Technique	3	0.37	.7753
Alignment × Skew	1	11.02	.0009
Technique × Alignment × Skew	3	0.94	.4229

Table 5.4: Repeated Measures Analysis of Variance for completion time against technique, alignment, and skew with interactions. DF Denominator is 1034 for all variables (1083 data points included).

The participants were asked this question without being informed of their relative performance with the various techniques. Out of the 34 undergraduate participants, 22 preferred the symSpline method, while only 3 listed symSpline as their least preferred method. The single-mouse technique was preferred by 3 participants and least preferred by 14 participants. Of course, there may be some bias in these numbers due to the 'good participant effect', despite efforts to prevent such a bias. Another interesting result is that of the two asymmetric techniques, the asymmetric right was more preferred than the asymmetric left, and the asymmetric left was the least preferred by many participants. This was expected after the pilot testing, although it appears to violate Guiard's model of asymmetric interaction. It may be that participants preferred moving the point with their right hand because it is an interaction they are accustomed to.

5.6 Spline Manipulation Conclusion

The symSpline technique was developed based on the guidelines of the Symmetric Interaction Model. It fits within the symDraw application and allows easy exploration of spline curves for novice users by addressing the three major problems with traditional spline manipulation.

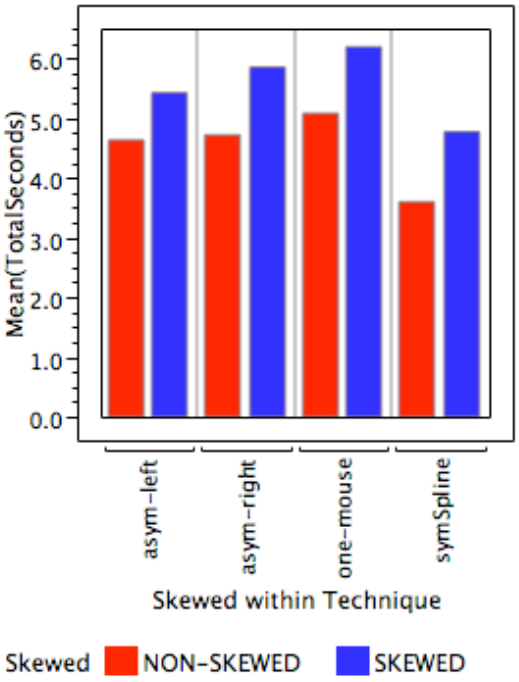


Figure 5.16: Mean trial completion times for skewed versus symmetric splines for each of the four techniques.

symSpline users don't have to choose what type of spline manipulation tool to use – there is only one. There is no need for mode-switching between different tools. And, because curve exploration is easy, users will be able to quickly gain intuition of how to manipulate the splines.

The experiment described in this chapter proves that symSpline is an effective technique for the manipulation of spline curves, at least in the simplified setting tested. New symSpline users are able to manipulate and match splines faster than with any of the other techniques tested. The symSpline technique proved to be especially beneficial for manipulating the more complicated S-curves. The participants were enthusiastic about the symSpline technique, preferring it over the other techniques in 22 of 34 cases.

The results from the experiment support the hypothesis of this thesis, that there exist symmetric tasks (such as locally editing curves) for which a symmetric interaction technique will outperform single-device techniques and dual-mouse asymmetric techniques. Chapter 6 will present a different application and set of techniques that were developed based on the Symmetric Interaction Model.

Technique	Most Preferred	Least Preferred
Single Mouse	3	14
Asym Left	4	13
Asym Right	5	4
symSpline	22	3

Table 5.5: Number of participants who declared each technique as their most preferred and least preferred.

Chapter 6

Image Processing

A symmetric technique for manipulating spline curves was presented in Chapter 5. Quantitative and qualitative evaluations showed that the symmetric spline manipulation technique generated faster performance and higher user satisfaction in a spline matching task. In this chapter, I present a set of bimanual symmetric techniques for manipulating digital images. While one of the techniques presented here is also based on curve manipulation (the tone-reproduction curve tool), it differs from the symSpline technique in that manipulation of the curve is the means to an end. With the symTRC tool that is presented in this chapter, the user is manipulating a curve to improve the underlying image. This level of indirection is significant because it shows that symmetric interaction can be used to manipulate non-spatial data.

The other image manipulation tools that will be presented in this chapter are also indirect. Both the ToneZone tool and the symLens tool are geometric figures that can be manipulated. However, the purpose of the manipulation is to improve the image or examine the image to evaluate the improvements, not to manipulate a geometric object. This chapter will present the symTone application that includes the symmetric image manipulation techniques, as well as details about the experimental analysis of the symTRC tool.

6.1 Previous Work in Tone-Reproduction Curves

Previous work in the area of tone-reproduction curves has focused on automatic tone correction for display device calibration, and has not involved human interaction with the curves. One recent project has used tone-reproduction curves for calibrating backlit-LCD screens for better

image range and lower power consumption[1] [IP05]. Stone, Cowan and Beatty give an excellent overview of the gamut-mapping process from traditional CRT to digital offset printing and the role of tone-reproduction curves in that process [SCB88]. Other work on tone-reproduction has focused on real-time tone-mapping for complex radiosity algorithms and high dynamic range imaging [GH97, DCWP02, RSSF02, ABWW03]. While some of this work does involve human interaction, the effectiveness of this interaction with either single-mouse or dual-mouse interfaces has not been studied. To my knowledge, there are no published experimental studies on human-computer interaction techniques for tone-reproduction curves, despite the fact that tone-reproduction curves have been available to the general public for use in image editing software for many years.

6.2 symTone

Figure 6.1: Comparison lens and file-related functions can be accessed via a pop-up transparent menu.

The symTone application is a symmetric image manipulation program designed to help

[1]Tone-reproduction curves are used to minimize distortion between the perceived brightness values of the individual pixels in an original image and those of the backlight-scaled image.

novices improve the quality of their digital photographs. Both grayscale and colour digital images can be manipulated using two mice in a way that allows easy exploration of the tone space. The symTone application provides three main functions: the ability to manipulate the tone-reproduction curve, the ability to manipulate the input and output tone ranges, and the ability to compare the modified and original images in detail in any area of the image. These three functions are described in detail here. Lens tool selection and file operations are accessible from a pop-up, radial, transparent menu that is launched with a middle mouse button click (Figure 6.1).

6.2.1 Tone-Reproduction Curve Manipulation

A tone-reproduction curve (TRC) is a common tool for mapping input tones to output tones in photographic manipulation. Applications such as Adobe Photoshop, the Gimp and PaintShop Pro all offer some version of TRC editing, usually labeled 'Curves'.

The TRC tool available in Adobe Photoshop serves as an example of how TRCs work (Figure 6.2). A default TRC is a straight diagonal line with a slope of 1, where each input tone in the tone range maps to an identical output tone. Output tones are adjusted by manipulating the TRC. For example, to lighten the mid-tones in a digital photograph, the TRC must be adjusted as shown in the middle image of Figure 6.2, where input tones in the middle range map through the TRC to tones that are lighter in the output range. A more complicated example might involve attempting to darken the dark areas while lightening the mid-tones and light areas. In this case, a TRC as shown in the bottom image of Figure 6.2 will achieve the desired result.

The Photoshop Curves tool allows the user to pick any point on the TRC and move the curve, and then pick more points to adjust the curve further, as shown in Figure 6.3. The difficulty with this tool is that multiple points can unnecessarily complicate the manipulation. In addition, because only one point can be moved at a time and the image does not update interactively, it is time-consuming and tedious to explore the many possible configurations of the TRC and their effect on the image being manipulated. Although expert users may know how to manipulate the TRC quickly to achieve the desired effect, novice users are much more likely to abandon the tool because it is difficult to use. It is also not obvious how to remove control points from the curve, or that they can be removed at all.

Manipulating TRCs in symTone provides four improvements over the technique available in Adobe Photoshop. The most important improvement is that users adjust the TRC symmetrically using two mice and two cursors. The user acquires two points on the curve and then moves the points around to adjust the curve, as shown in Figure 6.4. This allows a much more fluid exploration of the possible TRC configurations. The second improvement is that the image updates in real-time, while the user is adjusting the curve. The third improvement is

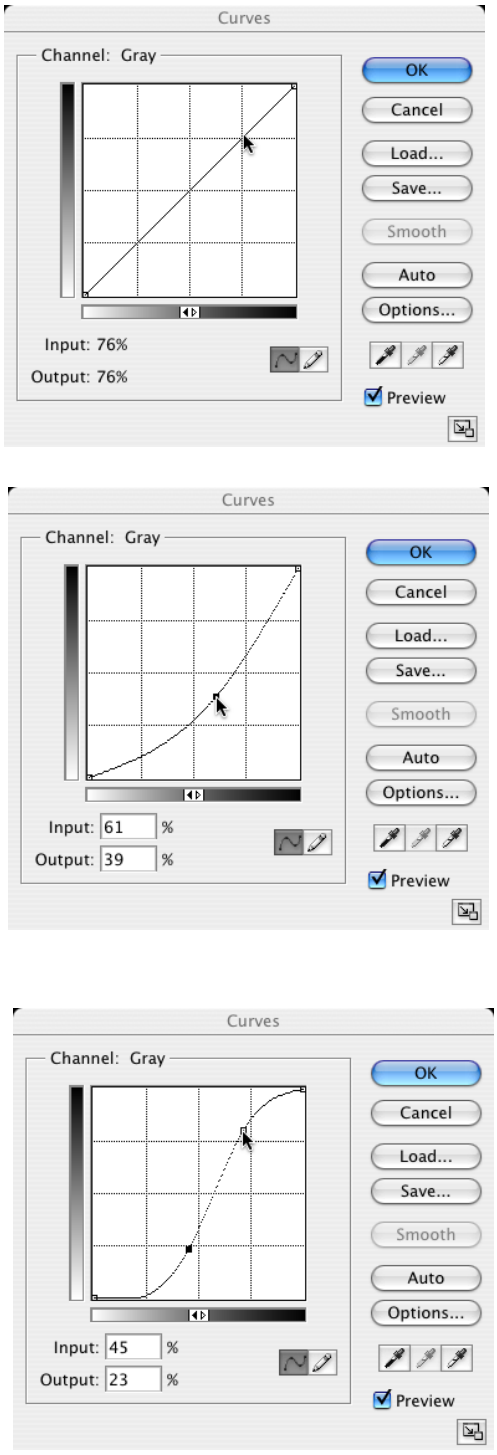


Figure 6.2: Tone-mapping maps each input tone through the TRC to an output tone, as shown in these screenshots from Adobe Photoshop. The top image shows a TRC in a neutral state, where each input tone maps to an identical output tone. The middle image shows the TRC pulled down to lighten the mid-tones of an image. The bottom image shows a more complicated mapping may lighten the light and mid-tones, while darkening the darker tones.

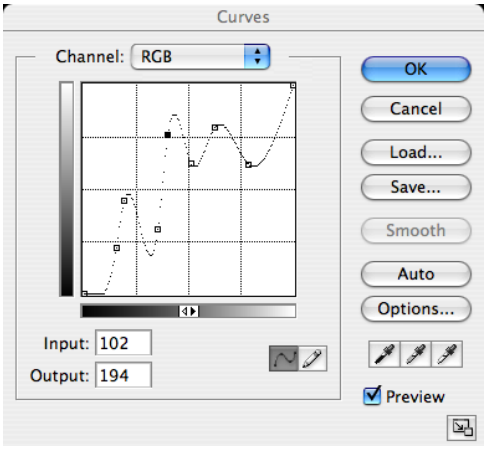


Figure 6.3: The curves tool in Adobe Photoshop allows users to add many points to the curve, which can become very complex.

the overlaying of the TRC on the image being manipulated. This means that users' attention need not be divided between the TRC widget and the photograph.

The fourth improvement over the traditional TRC widgets is that the TRC in symTone is represented by a single-segment Bézier spline. This means that additional control points are not added to the spline when the user clicks on it, and thus the curve cannot become too complicated to control. This has the added benefit of preventing the creation of non-functional curves with undefined relationships between the input and output tone ranges (which can occur in Photoshop). Although the use of a single-segment Bézier spline reduces the ability to have very fine-grained control over the tone-mapping, this is not usually required and is likely a worthwhile tradeoff.

Novice users are likely to find the symmetric TRC tool easy to use because the manipulation of the curve allows fluid exploration of the tone-mapping function. They do not need to know what the curve should look like in order to improve their particular image, they can simply move the curve around until they get the result they want. Over time, this may translate into knowledge of how the curve needs to be manipulated to generate certain effects.

6.2.2 The ToneZone Tool

The second main function provided by symTone is the ability to symmetrically and simultaneously manipulate the input and output tone ranges. Input tone range manipulation is commonly used in conjunction with a histogram to eliminate tones that are not used in the image, allowing more precise tone representation in the remaining tone range. Output tone

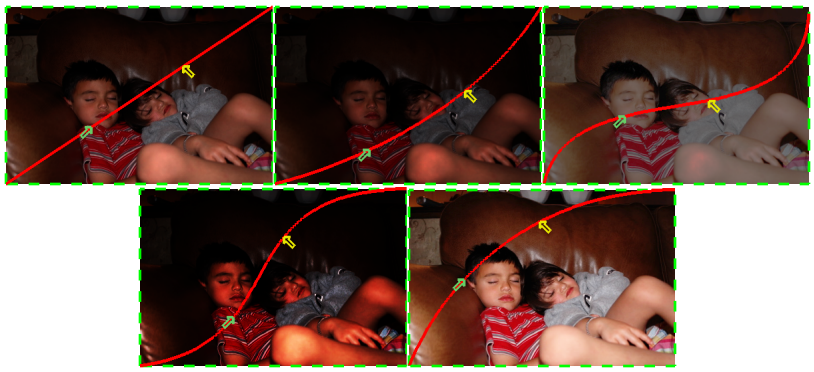


Figure 6.4: Using the two cursors in symTone allows users to quickly explore many different tone-mapppings.

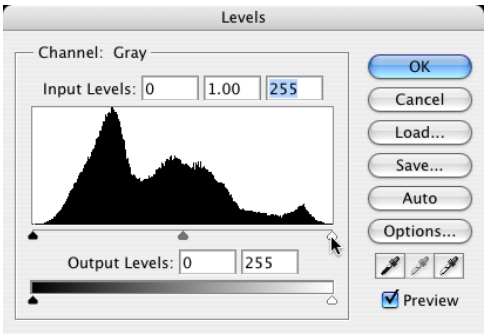


Figure 6.5: The levels tool in Photoshop is complex and presents five different handles for users to manipulate.

range manipulation is commonly used to adjust brightness and contrast levels in a digital image. These two functions are usually provided by discrete slider controls and are often labeled 'Levels' and 'Brightness/Contrast'. Figure 6.5 shows the 'Levels' tool in Adobe Photoshop where a user can adjust both the light and dark ends of the input and output tone ranges. Figure 6.6 shows the 'Brightness/Contrast' tool being used in Adobe Photoshop.

The difficulty with adjusting input and output tone ranges in typical applications is the discretization of the four degrees of freedom: light input tone level, dark input tone level, light output tone level and dark output tone level. Because these variables are manipulated individually, getting all four of them in the positions necessary for a desired photographic result is time-consuming and tedious. As in the TRC manipulation, expert users may know exactly which of the four variables needs to be adjusted and in which direction. However, a novice user is not likely to know which adjustments to make (or in which direction) and may

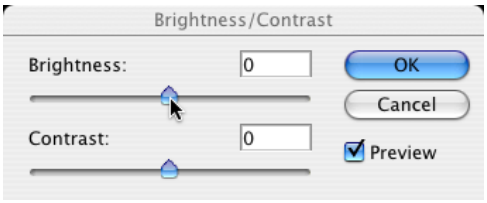


Figure 6.6: The brightness and contrast tool in Photoshop presents two separate sliders for the user to manipulate.

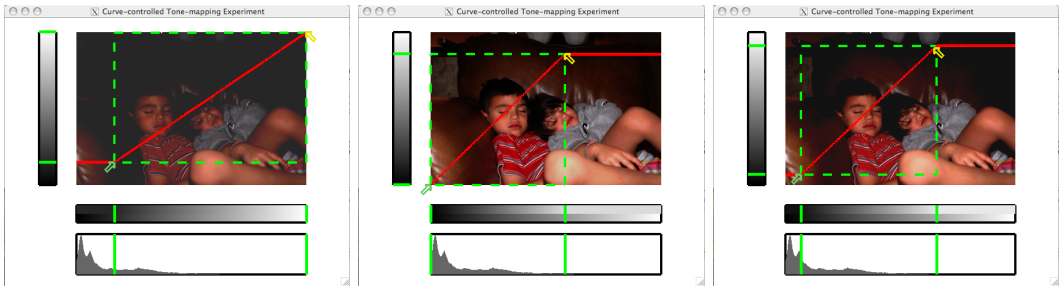


Figure 6.7: The ToneZone Tool. Adjusting the lower-left corner of the rectangle limits the black tones in the input and output ranges. Adjusting the upper-right corner of the rectangle limits the white tones in the input and output ranges. By moving both cursors, the user can explore any combination of input and output tone range boundaries.

abandon these tools in frustration.

The ToneZone tool in symTone combines the adjustment of these four variables into a single, symmetric manipulation. By using two cursors to control the position and size of a rectangle, the input and output tone ranges can be specified (Figure 6.7). Because the user can translate and scale the rectangle by moving the cursors around the screen, there is no mode switching required. The horizontal position and width of the rectangle determines the size of the input tone range. A rectangle that is further to the left will clamp tones at the light end of the tone-range, while a rectangle that is further to the right will clamp tones at the dark end of the tone-range.

The vertical position and size of the rectangle specify the output tone range. A rectangle that spans the entire photograph from top to bottom includes the entire output tone range. A rectangle that is closer to the bottom of the photograph eliminates the lightest output tones, causing the image to be darker. A rectangle that is closer to the top of the photograph eliminates the darkest output tones, causing the image to be brighter. The shorter the rectangle, the lower the contrast in the image because the selected range of input tones is mapped to

a smaller set of output tones. Making the rectangle more narrow (horizontally) increases the contrast of the image because a smaller set of input tones is mapped to the selected range of output tones.

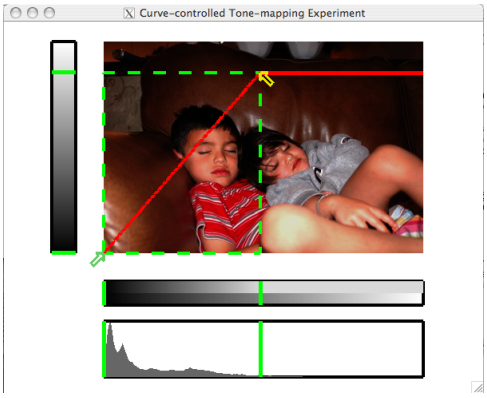


Figure 6.8: Intermediate and expert users can make use of the histogram when manipulating the range-reduction rectangle to limit the input tones to those that are salient.

As the user manipulates the ToneZone rectangle, green boundary lines are superimposed on the input and output tone range bars at the bottom and left of the image (Figure 6.7). These indicators align with the user's cursor positions, showing what is being manipulated by the rectangle size and position. Additionally, a histogram showing the salience of the tone range in the image is displayed at the very bottom of the symTone screen. Green indicator lines are also shown on the histogram, allowing expert users to manipulate the input tone range in a familiar fashion (Figure 6.8). In some graphics applications, moving the levels indicators causes the histogram to resize so that the selected minimum and maximum input tone ranges are at each end of the histogram. The symTone ToneZone tool does not do this dynamic resizing.

The user is able to invert the image by vertically flipping the rectangle, so that the left cursor is above the right cursor (Figure 6.9). This is sometimes a desirable feature for artists who want a 'solarized' version of an image. Users cannot horizontally flip the rectangle as this would create an image with no input tones.

As with the TRCs, a novice user does not need to know which input and output ranges are required to give them the desired image. Because all four variables are manipulated simultaneously while the user moves the rectangle around, the user can explore the entire tone range domain quickly and easily. This means users can move the rectangle until they get the desired result. Because the tone range rectangle is superimposed on top of the image being manipulated, the user does not need to divide his attention between the ToneZone tool and the image being manipulated.

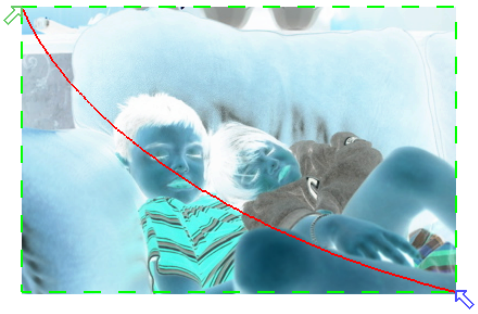


Figure 6.9: Inverting an image by vertically flipping the range rectangle.

It should be noted that the TRC manipulations are maintained within the ToneZone tool and vice versa (Figure 6.10). Thus, if the user switches from TRC manipulation to ToneZone manipulation, the shape of the TRC is maintained (and displayed) within the ToneZone. The user can switch back to TRC manipulation and the mapping adjustments will apply to the reduced input and output ranges. The right image in Figure 6.10 shows that while manipulating the TRC, the range indicators are still displayed on the input and output display bars. This allows the user to see the input and output tone ranges that will be affected by the TRC adjustment.

The ToneZone with a Single Mouse

While the ToneZone tool was developed based on the Symmetric Interaction Model and is designed to work with two cursors and two mice, it is obvious that the ToneZone could be used as a single-mouse widget. I developed a prototype single-mouse version of the ToneZone tool. It requires the user to select either the top-right or bottom-left corner of the ToneZone for manipulation. The user has to switch back and forth between these two controls, so it is clearly inferior to the dual-mouse symmetric version (which requires no switching). However, it does allow the user to control the four variables (input and output minimum and maximum tone levels) using two controls, rather than four, so it is likely to be superior to the traditional single mouse techniques for manipulating these variables.

The single-mouse ToneZone widget is shown in Figure 6.11. When the mouse button is not pressed, the ToneZone rectangle is invisible except for two corner indicators, so that the user knows where to reacquire the ToneZone controls. An interesting feature that was added to the single-mouse ToneZone is the ability to move the whole rectangle around by clicking in the middle and dragging, as shown in Figure 6.12. This allows the user to control all four variables at once. Testing and evaluation of this single-handed version is certainly an area for future investigation.

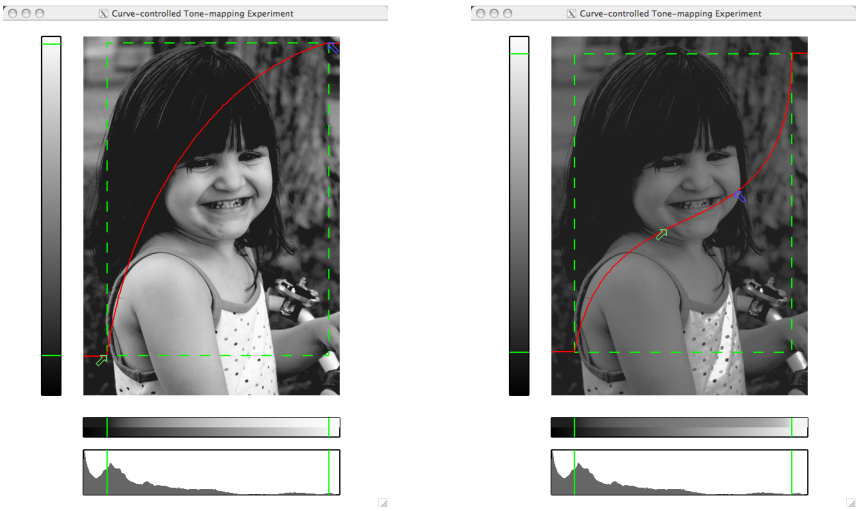


Figure 6.10: The screenshot on the left shows that the tone-range rectangle can be manipulated after the tone-mapping curve has changed. The screenshot on the right shows that the tone-mapping curve can be adjusted after the tone-range rectangle has changed.

While the single-mouse ToneZone has not been formally evaluated, its development is evidence that new dual-mouse techniques can be developed based on the Symmetric Interaction Model and these techniques may in turn lead to new single-mouse techniques that would not have been developed otherwise.

6.2.3 Version Comparison

The third function provided by symTone is the ability to compare details of the manipulated image with the original version in any area of the image. This is achieved through the use of a symmetrically controlled 'version lens'. The user controls the position and size of a circular lens that can be moved over the original photograph. The image inside the lens is the modified version, while the image outside the lens is the original. Figure 6.13 shows the version lens in use. Once the user has finished their comparison, they can accept the modified image by dismissing the version lens or they can revert to the original image. This technique was inspired by the MagicLens widget and by the Pie Versions widget.

6.2.4 The Aspect Ratio Effect

By superimposing both the tone-reproduction curve and the tone range manipulation rectangle on top of the image being manipulated, the issue of dividing the user's attention between

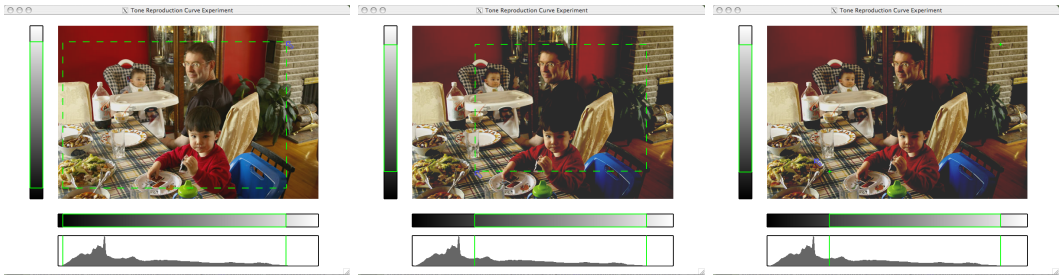


Figure 6.11: Manipulating the ToneZone with a single mouse and cursor.

Figure 6.12: Dragging the ToneZone with a single mouse and cursor.

widget and data is dismissed. However, there is a potential problem related to superimposing the widgets onto the image. This is related to the aspect ratio of the image being manipulated. If the image is square, then the same level of visual detail is provided for both the input and output tone bars. However, if the image is strongly rectangular (i.e. a wide landscape size, or a tall portrait size), the tone bar on the short side of the image is going to provide less visual detail than the tone bar on the long side of the image.

In addition, the manipulation will be affected by the aspect ratio. For example, the user will need to manipulate a taller TRC in a portrait photo and a shorter TRC in a landscape photo. For tone range manipulation, the user will have more precision with which to manipulate the output tone ranges for portrait-sized images and more precision with which to manipulate the input tone ranges for landscape-sized images. While this may be a limitation of the design, the ease of exploration of the domain space as a whole likely outweighs this disadvantage.

One way to address this issue is to make the manipulation widgets square, and centered over the image. However, this may detract from the unified aspect of the interface: in the current design the user can maximize the ToneZone easily by making the rectangle take up the entire space of the image, and the ToneZone rectangle then blends into the edge of the image. If a square ToneZone were superimposed on the image, the rectangle would not blend into the image when maximized. This could give the user the impression that the part of the

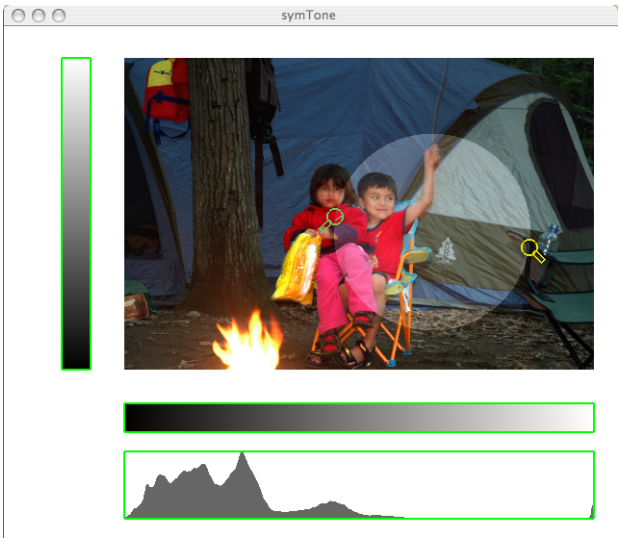


Figure 6.13: Comparing the original image (inside the circle) with the altered image (outside the circle) using the symTone lens.

image outside of the ToneZone is unaffected by the manipulations.

6.3 Experimental Setup

A formal experiment was conducted to test the symmetric TRC technique against other TRC manipulation techniques. The symmetric technique was tested against both an asymmetric dual-mouse technique and a single-mouse technique.

6.3.1 Task

The experiment involved a tone-matching task with simple graphics. For each trial, the participant had to manipulate a tone-reproduction curve such that the tones of four greyscale squares in an image matched the tones at the border and center of the image (Figure 6.14). The tones at the border and in the center diamond were overlays which were unaffected by TRC manipulations. The TRC manipulations affected only the tones in the four quadrants. The tone bars that are present in symTone at the bottom and left of the image were also present in all three conditions for the experiment.

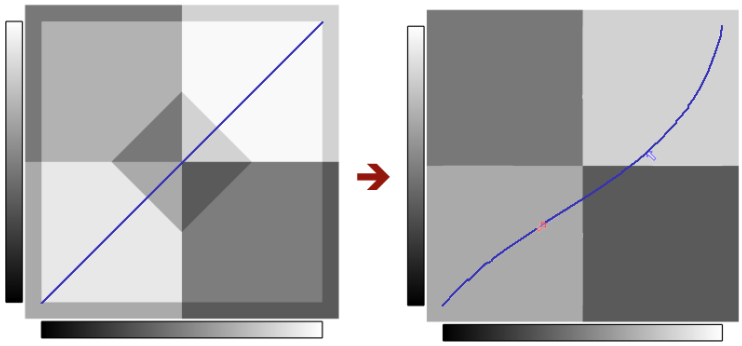


Figure 6.14: Trial task. Participants were asked to adjust the tones in each of the four quadrants so that they matched the tones in the center diamond and at the border. The image after TRC adjustment is shown at the right.

6.3.2 Techniques

Three different techniques were tested as conditions in the repeated measures experiment. These techniques are summarized in Table 6.1 and described in detail here. For all three techniques, only one or two points on the TRC could be manipulated. These points could slide along the curve, allowing the user to create a broad range of TRCs. This is different from typical TRC widgets where an arbitrary number of points can be added.

SING Technique

For the single-mouse (SING) technique, participants manipulated the TRC using the right mouse only. Pressing the inner (left) button allowed the user to manipulate the left point on the TRC. Pressing the outer (right) button allowed the user to manipulate the right point on the TRC. The user was not able to manipulate both points together. When the user pressed down the inner button, the blue arrow cursor warped to the left point and when the user pressed down the outer (right) button, the red arrow cursor warped to the right point. Thus, there was no need to *acquire* the points on the TRC. The SING technique was loosely based on traditional TRC manipulation techniques, where the user picks a point and moves it.

ASYM Technique

For the dual-mouse asymmetric (ASYM) technique, participants used one mouse to manipulate a point on the TRC and the other mouse to adjust the tangent at that point. By pressing the inner button on the left mouse, the participant could move a point on the TRC. By pressing

Technique	Description
SING	Use right mouse only Drag left point pressing left button Drag right point pressing right button
ASYM	Use both mice Drag point with left mouse, pressing right button Drag tangent with right mouse, pressing left button
SYM	Use both mice Drag points with two mice, pressing inner buttons

Table 6.1: Techniques tested in experiment. 'Inner' button refers to the left button on the right mouse and the right button on the left mouse.

the inner button on the right mouse, the participant could adjust the tangent and therefore the angle of the TRC (at the point controlled by the left mouse). The participant could adjust both position and angle simultaneously. When the inner button on the left mouse was pressed, the left cursor warped to the point on the TRC. When the inner button on the right mouse was pressed, the right cursor warped to the end of the tangent. This technique is similar to the asymmetric technique evaluated in the splines experiment described in Chapter 5. I did not test the opposite asymmetric technique (i.e. where the right mouse would move a point and the left mouse would adjust the tangent) because the two asymmetric techniques in the splines experiment in Chapter 5 did not generate statistically significant results. Testing a fourth technique would have either caused the experiment to run longer, or would have allowed fewer data points to be collected for each technique.

SYM Technique

The dual-mouse symmetric (SYM) technique worked as outlined in the description of the symTone application. The user pressed the inner mouse buttons and dragged two points on the curve to adjust the TRC. Releasing one of the mouse buttons allowed participants to adjust the curve moving the point controlled by the other mouse. When both mouse buttons were released, the curve was static. The points on the curve did not have to be acquired: when the user pressed down the inner button on a mouse, the cursor controlled by that mouse was automatically warped to the appropriate point.

6.3.3 Test Images

Each trial image consisted of a simple set of 4 greyscale squares. These simple graphics were chosen instead of real photographs to avoid distracting the participants. Four target tones

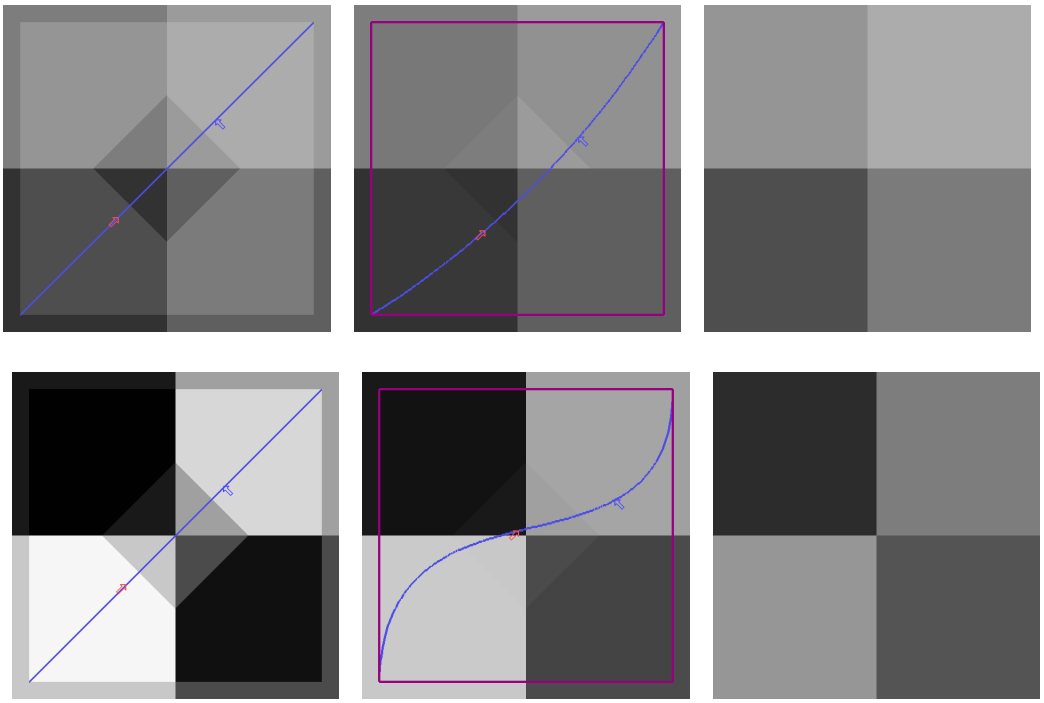


Figure 6.15: Two sample input images, with final curve images and target images. For each input image (left), the participant had to manipulate the TRC (middle) to make the image match the target (right).

were shown at the border of these squares and in a diamond superimposed on the image. An example input image is shown in Figure 6.14. There were five different target tone sets and for each of these five target sets there were four input images. Thus, there were 20 input images in total. Two samples are shown in Figure 6.15.

6.3.4 Design

The experiment was a within-participants design. Twenty-one undergraduate students served as participants and were paid $10 each for participating. The order of conditions was counter-balanced. The participants were broken into six groups, so that each combination of condition orders was seen by three or four participants. The experiment took approximately one hour to complete. The participants were given time to practice with all three techniques before completing timed trials. In particular, because the experiment task was quite difficult, each

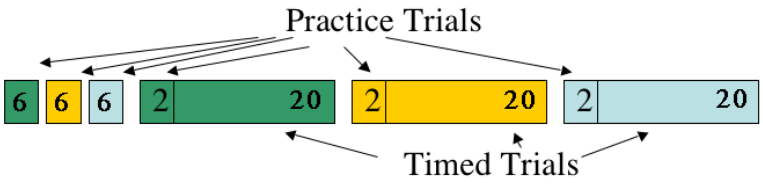


Figure 6.16: Participants performed 6 practice trials for each condition, then performed 2 practice trials followed by 20 timed trials for each condition.

participant was given six practice trials for each of the three conditions at the beginning of the experiment. This was then followed by three blocks (one per condition), where each block consisted of two practice trials and 20 timed trials (Figure 6.16).

The metric in this experiment was the trial completion time, measured from the appearance of the tones until the tones have been matched. The match threshold was measured by comparing the tones of the four quadrants (ranging from 0-255) with the four target tones (also ranging from 0-255): the total error could not exceed 40.0. If this error was spread evenly across all four quadrants, the differences between the input tones and the target tones would be difficult to perceive.

6.3.5 Hypothesis

The main hypothesis of this experiment was that the symmetric technique would have the lowest average trial completion time, and the single mouse technique would have the highest average trial completion time, with the completion time for the asymmetric technique falling in the middle.

6.4 Results and Discussion

Not all participants were able to finish the experiment. Three of the 21 participants (participants 3, 8 and 16) found the task too difficult, and quit after the initial practice trials. These three participants tried the six practice trials for all three techniques, but this took them an excessive amount of time (more than 40 minutes). It is unclear why these participants had so much difficulty with the task. Participant 3 reported that she simply could not do the task, regardless of the technique. Participant 8 reported that she liked the SYM technique better, but even using that technique, she found the task too difficult. Participant 16 reported that she preferred the SING technique, but found the task too difficult with all three techniques. As there are no timed results from these three participants, their data are not included in the analysis that follows.

6.4.1 General Results

Data for 18 participants were collected. Each participant performed 60 timed trials, for a total of 1080 trials. 45 outlier trials were removed from the data, being greater than two standard deviations from the mean trial time, leaving 1035 trials for analysis. As there were no timeouts in the experiment program, the outliers are likely results of trials in which a participant got distracted or frustrated and stopped temporarily in the middle of a trial. The outlier trials included trials from all three conditions and accounted for less than 5% of the data, which was acceptable, given the difficulty of the task.

Over all participants, the SYM technique had the lowest mean trial completion time, supporting the hypothesis. The SING technique had the second-highest trial completion time, and the ASYM technique had the highest trial completion time. However, the difference between the SING and ASYM techniques is negligible. The mean times for the three techniques are shown in Table 6.2.

Technique	Mean Time	Std Dev
SING	16.84	4.29
ASYM	16.87	3.08
SYM	14.93	4.34

Table 6.2: Results over all trials. Times are in seconds.

An analysis of variance was performed to determine the statistical significance of the performance differences between the three techniques. Table 6.3 gives the F-ratio for condition of 3.6, $p_{2,28} = 0.0392$, which shows that the effect of the condition variable is statistically significant. Additionally, post hoc tests were done to analyze the pair-wise differences between the three conditions. These results are shown in Table 6.4.

	Variable	DOF	F-ratio	Prob
	Constant	1	473.76	≤ 0.0001
1	condition	2	3.64	0.0392
2	block	2	2.75	0.0812
3	participant	17	4.90	0.0001
4	cond $\times$ block	4	1.14	0.3571
	Error	28		
	Total	53		

Table 6.3: Analysis of Variance for completion time against the following variables: condition, block and participant (random).

The analysis of variance shows that the block variable, which represents the first, second or third block of trials for each participant is not significant (Line 2 in Table 6.3). The interaction between block and condition was also included in the ANOVA, but was not statistically significant (Line 4 in Table 6.3).

Participant was included as a random variable in the analysis of variance and represents the 18 participants whose data was analyzed. This variable is statistically significant (Line 4 in Table 6.3), which shows that there is a high variance in the trial completion time across the participants. This was not surprising given the wide variance in time taken to complete the experiment – ranging from 40 to 75 minutes.

Pair	Difference	Prob
SING - ASYM	-0.03	0.96
SYM - ASYM	-1.94	0.025
SYM - SING	-1.91	0.028

Table 6.4: LSD post hoc tests for condition. Difference is in seconds.

The post hoc tests shown in Table 6.4 show that the difference between the SYM condition and the ASYM condition is significant, as is the difference between the SYM condition and the SING condition. However, there is no statistically significant difference between the AYSM and SING conditions. Thus, our hypothesis is partially supported, in that participants were significantly faster with the symmetric technique than with the other two techniques. However, we hypothesized that the asymmetric technique would be faster than the single mouse technique, and there is no significant difference.

6.4.2 Subjective Results

Each participant was asked to fill out a NASA Task Load Index (TLX) [HS88] evaluation of the task after performing the 20 timed trials for each of the three conditions. The NASA TLX is a standardized questionnaire used in HCI for evaluating the physical and mental load placed on a user performing a task. The questionnaire is included in Appendix C. The responses from these evaluations are shown as averages over the 18 participants in Figure 6.17. These results show that the participants found the symmetric technique to require the least amount of effort, to produce the least amount of frustration and to be the least mentally demanding of the three techniques. However, the participants did find the symmetric technique to be more physically demanding than the single mouse technique. The higher physical effort for both the dual-mouse techniques compared to the single mouse technique is not surprising as none of the participants were accustomed to using a mouse with their left hand. The fact that participants found the symmetric technique to be the least mentally demanding, in spite of not being accustomed to using a mouse in their left hand, is very encouraging.

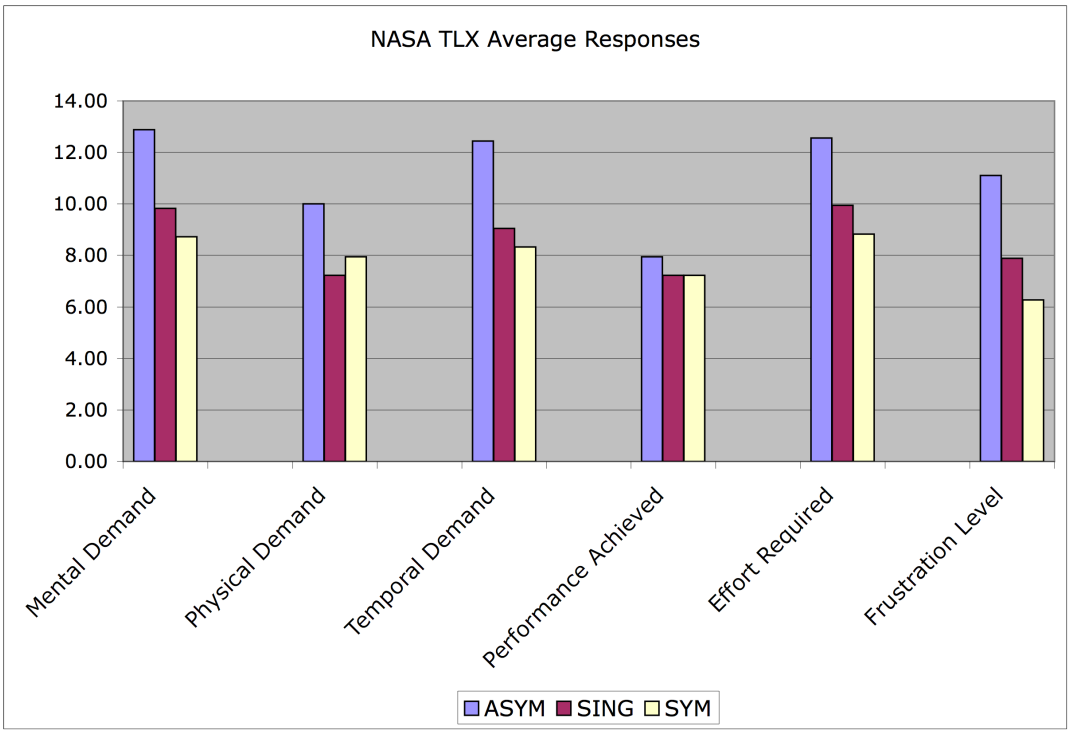


Figure 6.17: Averaged responses of 18 participants to NASA TLX questions, for each of the three conditions.

The NASA TLX questionnaires can also generate a total workload calculation. The formula for this calculation takes the numeric rating for each category (0-20) and multiplies that number by the number of times that category was chosen on the task demand questionnaire (the second page). This product is calculated for each of the six categories, summed and then the total is divided by 200. The final workload figure is a number between 0 and 100. The middle column of Table 6.5 shows the total workload for each of the three conditions, averaged across participants. The right column of Table 6.5 shows the total workload without the physical demand product. If dual mice were part of the standard desktop and people were accustomed to using two mice, they would be unlikely to consider the dual-mouse conditions more physically demanding. Thus, the right column of Table 6.5 shows what the total workload calculations might look like in a standard, dual-mouse world. In both cases, the symmetric condition is rated with the lowest workload.

In addition to the NASA TLX, we asked participants to rank the three techniques in order of preference after they completed the experiment. The participants were asked, "If you had a job matching image tones all day, which of the techniques would you most prefer to use

Mode	Total Workload	Total Workload (physical component removed)
ASYM	60.86	56.69
SING	46.76	44.49
SYM	44.16	39.67

Table 6.5: NASA TLX total workload (with and without physical workload component) for each condition, averaged over 18 participants.

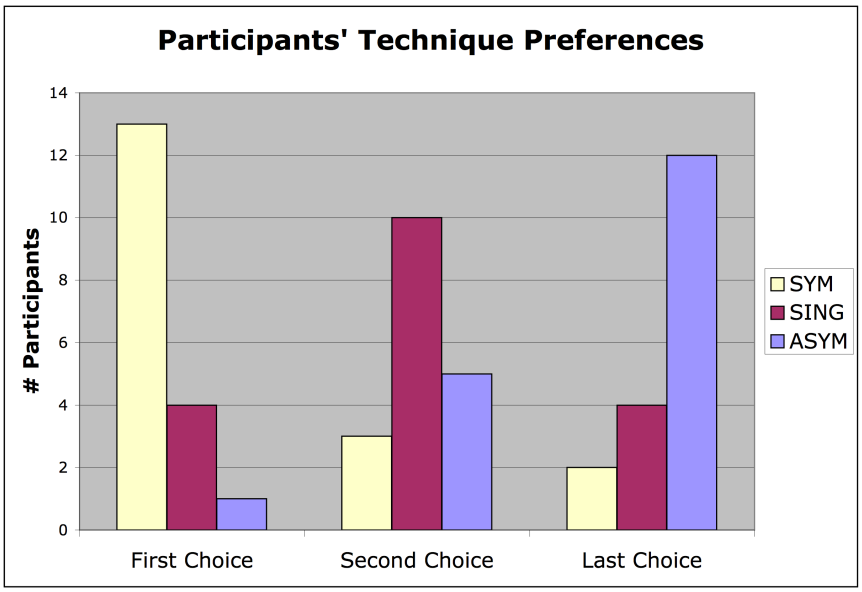


Figure 6.18: Participants' technique preferences, from post-experiment debriefing.

and least prefer to use?" Of the 18 participants who completed the experiment, 13 ranked the symmetric technique highest, four ranked the single mouse technique highest and one ranked the asymmetric technique highest. The preference rankings are shown in Figure 6.18.

6.4.3 Discussion

The symmetric TRC manipulation technique yields small but significant performance gains over the single mouse and asymmetric dual mouse techniques. The empirical and subjective results support the idea that manipulating a curve with two hands is a task that is more naturally performed symmetrically. This confirms the findings from the symSpline experiment in Chapter 5, although the symSpline technique is different than the symTRC technique pre-

sented in this chapter. However, the real benefit of the symmetric TRC technique is likely in its expressiveness.

Alternative Tests

The symmetric TRC manipulation technique allows novice users to explore the tone-mapping domain quickly and without a priori understanding of how tone-mapping works. An experiment based on timed results is unlikely to fully demonstrate the benefits of this expressiveness. It is likely that an experiment based on testing the range of options explored would more fully demonstrate the benefits of the symTRC technique over TRC techniques found in traditional graphics applications. Such an experiment would have to be based on a fixed time and could not be based on users matching tones.

Usability Improvements

The symTone TRC technique also has other benefits over typical TRC widgets that were not tested by this experiment. The simplicity of the cubic Bézier spline to which control points cannot be added is one of the main benefits for novice users. The fact that the curve is superimposed over the image avoids the problem of dividing the user's attention between a widget control and the image being manipulated. Additionally, the symTone TRC is fully interactive – the image updates immediately as the curve is manipulated. The curves tool in Adobe Photoshop claims to be interactive but is sluggish – updates occur when the mouse is idle. The curves tool in the Gimp is even less interactive: the current tone-mapping is applied to the image *after* the mouse button is released. These interface improvements, in combination with the basic interaction performance gain shown in our experiment results and the participants' TLX scores, demonstrate the usability and effectiveness of the symTone TRC technique.

6.5 Summary

The symTone application is an example of symmetric interaction techniques applied to a non-geometric domain. The TRC experiment supports the Symmetric Interaction Model by showing another example of a symmetric technique for a symmetric task with better performance and higher user satisfaction than a single-mouse technique or an asymmetric dual-mouse technique. The ToneZone widget, although not formally evaluated for this thesis, has obvious performance gains over a traditional single-mouse technique due to the fact that the user requires one control rather than four. It is also important to note that a single-mouse version of the ToneZone can be used and would require two controls rather than four. ToneZone is

an example of a new technique developed because of the availability of the symPut driver and based on the Symmetric Interaction Model, but that can be applied in a single-mouse context and provide potential performance benefits.

It seems likely that symmetric two-handed interaction could be useful in other ways in the image editing domain. Symmetric curve manipulation may be useful for editing high dynamic range images in OpenEXR format. There might also be performance gains in using gradient selection. These are all good areas for future investigation.

Chapter 7

Conclusions

This thesis has presented both a Symmetric Interaction Model and examples of techniques and applications that have been built based on it. A model for symmetric interaction is needed in the HCI literature to complement Guiard's Kinematic Chain Model for asymmetric interaction [Gui87]. While there are many everyday tasks that the Kinematic Chain Model applies to (such as handwriting), it does not apply to everyday tasks such as clapping, skipping rope, shuffling cards or typing at a keyboard. Similarly, there are a number of tasks in the human-computer interface where the Kinematic Chain Model does not apply, because the tasks are naturally symmetric.

7.1 The Symmetric Interaction Model

Most naturally symmetric tasks in the user interface have been artificially broken into subtasks so that they can be performed in serial by a user with a single mouse and cursor. These tasks (such as window manipulation) could benefit from a symmetric interaction technique. The Kinematic Chain Model does not provide guidance on how to develop such techniques. The Symmetric Interaction Model presented in this thesis provides that guidance.

The Symmetric Interaction Model was summarized in Chapter 3 by the following four points. These four points are restated here and elaborated on with respect to the applications and techniques that were presented in Chapters 4, 5 and 6.

1. Symmetric interaction is a superset that includes asymmetric interaction. An interface that affords symmetric interaction can be used asymmetrically, but the reverse is not usually true. All of the techniques presented in this thesis afford the user the ability to work symmetrically, where the two hands contribute to the task equally. However, these

techniques also allow a user to work in serial, and do more of the interaction with their dominant hand, letting the non-dominant hand follow, if desired. In addition, symmetric techniques can be situated within applications that also make use of asymmetric and single-mouse interaction techniques.

2. Symmetric interaction requires symmetric devices. All of the techniques tested in this thesis were tested using identical three-button USB mice. I strongly believe that the symmetric techniques tested in the experiments in Chapters 4, 5 and 6 would not have generated such positive results if the two devices used by participants were different (such as a puck and stylus). The difference in control-display ratios and the different feel of the two devices would greatly decrease the symmetric feel of the interaction and would likely have led to degradation in performance. In the special case of a single device used for symmetric interaction (such as a large trackball or a multi-touch screen) the device must allow the two hands equal affordance.

3. Symmetric interaction requires bilateral function symmetry. Pressing a button with a given finger on one hand should activate the same functionality if the same finger on the other hand presses a button. The techniques presented followed this convention. There were no reports from experiment participants that this was awkward to use or difficult to get used to or suggesting a different mapping.

4. Symmetric interaction should involve a unified task, where the two hands work together to manipulate a single object. All of the techniques tested involved unified tasks, because they all involved manipulating some type of geometry with both cursors (lines, rectangles, splines or ovals).

In addition to these four points, there are other aspects of the Symmetric Interaction Model which have been supported by the techniques and experiments presented. There were no restrictions in which hand started or ended the interaction for the symmetric techniques presented, and that seemed to work well. In all of the experiments the two cursors were differentiated by colour. In the TRC experiment they were also differentiated by shape (left-pointing and right-pointing arrows). In informal evaluation, users commented that they liked the arrow cursors with different alignments and colours and so the arrow-shaped cursors were added to the symDraw application.

The Symmetric Interaction Model has also provided a taxonomy of symmetries that can be applied to the user interface. This should provide designers and researchers with a more precise vocabulary to use in discussing symmetry in the user interface. This is an important contribution because the word symmetry is so overloaded and can cause confusion because of its many different meanings.

7.2 The Bimanual Interaction Decision

This thesis has provided, as part of the Symmetric Interaction Model, guidelines for designers who want to implement two-handed interaction techniques for a given task. In particular, the first decision a designer should make is whether a symmetric or an asymmetric technique is more appropriate. This means that the designer has to determine whether the task is naturally symmetric or naturally asymmetric. The best test in making this decision is to try to determine an appropriate split of the task into asymmetric sub-tasks, where the non-dominant hand performs the 'easy' sub-task and the dominant hand performs the 'hard' sub-task. If there does not seem to be an easy/hard split, or it can't be decided which hand should do which task, then the task is likely symmetric and a symmetric technique should be developed. This rule of thumb proved to be true for the experimental tasks tested in Chapters 4, 5 and 6.

7.3 Non-spatial Applications

An important idea contributed to the HCI literature by this thesis is that symmetric interaction can be applied to non-spatial (i.e., non-geometric) tasks. The TRC experiment results reported in Chapter 6 demonstrate that using a symmetric interaction technique to manipulate non-spatial data (in this case, image tones) is effective. While the technique involved the manipulation of a spline curve, the curve was the means to an end, rather than an end itself. While this point was not part of the original Symmetric Interaction Model used while developing the techniques presented in the thesis, it is an important contribution.

7.4 The Applications

The symDraw and symTone applications provide examples of how to design applications that easily and gracefully afford symmetric interaction. Important aspects of these interfaces include:

No resident widgets on screen The lack of control widgets on screen in both applications has multiple benefits. First, there is more screen real estate available for the application data (the digital image in symTone or the artwork in symDraw). Second, there is no handedness bias relating to the positioning of widgets, as this would break the symmetry of the interface.

Transparent popup menus Using transparent menus that popup under the cursor and can be activated by either mouse helps to maintain the symmetry of the interaction. Users can invoke the menu with whichever hand they choose and can make menu item choices

with either hand. This helps to prevent handedness bias, and maintains accessibility to users across the handedness spectrum.

Asymmetric and single-mouse interaction Both symDraw and symTone allow the user to perform menu selections in either a symmetric or asymmetric way. Object selection (such as selecting a shape to edit in symDraw) is obviously a single-mouse technique, and can be performed with *either* mouse. The same is true for the creation of spline curves in symDraw, which can be sketched with *either* hand. These examples show that symmetric applications really do afford other types of interaction (again, supporting the idea that symmetric interaction is a superset that includes asymmetric interaction), allowing the user choice in how she uses the mice, and providing the best style of interaction for each type of task.

No keyboard shortcuts While keyboard shortcuts are standard tools in typical desktop applications, they are absent from both symTone and symDraw. This is because it is expected that the interaction in these applications will be entirely mouse-based. Of course, at some point a user might have to use the keyboard to type in a filename, and this thesis is not advocating that keyboards be thrown out. However, it is just as easy to take two hands off of two mice and move them to the keyboard to type as it is to take one hand off of one mouse and move it to the keyboard (the two hands can move to the keyboard in parallel). There is no extra time cost associated with this style of interaction. And, where keyboard modifier keys are used to add extra information or apply a mode in typical applications, the extra degrees of freedom available from the second mouse in symTone and symDraw can be used instead.

The symTone and symDraw applications can be considered a graphical application suite, because the interface and interaction techniques are consistent across the two applications, affording learning transfer.

7.5 The Techniques

Seven different symmetric interaction techniques were presented in this thesis. Four of these were formally evaluated with user experiments; the others were informally evaluated. These techniques and results are summarized here.

Object rotation and translation Although this technique was first implemented by Kurtenbach et al. [KFBB97], it was never formally evaluated. The evaluation performed in this research showed that while users might initially have difficulty with large rotations due to the flipping of the cursors, this difficulty quickly disappears. The symmetric technique outperformed asymmetric and single-mouse techniques after 2 hours of exposure to the

techniques. Users would probably avoid the flipping issue in a non-experimental setting by flipping the object halfway, releasing it and then reacquiring it in the new orientation.

Object rotation, translation and scale This technique was also implemented by Kurtenbach et al. [KFBB97], and never evaluated. The evaluation for this research took place in the context of an image registration application. The results clearly showed that the symmetric technique allows faster task performance than the dual-mouse asymmetric technique and the single-mouse technique. Additionally, the symmetric technique was clearly favored by experiment participants.

Colour selection The colour selection technique that is part of symDraw is symmetric in a different way than the object manipulation techniques. The colour selector allows the user to manipulate the position of a colour palette with one hand and the position of the selection cursor with the other hand. Either hand can do either task, and once a colour is chosen, the selection button-press can be performed by either hand. This technique can obviously be performed asymmetrically, with the non-dominant hand simply holding the palette in place and the dominant hand moving the cursor over the palette. However, it can also be performed symmetrically where the cursor and the palette both move together. Informal user testing with an artist showed that the artist initially picked colours asymmetrically, but over time moved to picking colours symmetrically. The colour selection technique is a special case of the general menu manipulation that is part of both symDraw and symTone. With all of these menus, users in informal testing initially forgot that they could move the menu while making a selection, but soon got used to that and took advantage of it.

symSpline The symSpline technique allows users to locally manipulate the area of a spline by using two cursors to control the ends of the tangent to a point on the spline. The formal experiment for the symSpline technique compared it to a single mouse technique as well as two dual-mouse techniques. The symSpline technique strongly outperformed the other techniques and was preferred by experiment participants.

symTRC The TRC manipulation technique in the symTone application is also a new technique that was developed based on the Symmetric Interaction Model. In formal user experiments, the TRC technique outperformed a single-mouse technique and a dual-mouse asymmetric technique. While the quantitative results of this experiment were not as strong as the results in the symSpline and image registration experiments, the qualitative results gathered using the NASA TLX index show that users prefer the technique, find it less frustrating and find that it requires less mental demand.

symLens The symLens technique is part of the symTone application. It allows users to interactively control the size and position of a circular lens superimposed on an image. The original image is inside the lens and the modified image is outside the lens. Because the

task performed by this technique is comparison, rather than modification, it is difficult to gauge the performance. However, informal comments on this technique have been strongly positive.

ToneZone The ToneZone tool is a new symmetric technique for which there is no single-mouse equivalent. The ToneZone tool replaces the contrast slider, brightness slider and the levels tool from typical graphics program with a widget that controls all of these parameters with a single bimanual gesture. ToneZone's strength is in its expressiveness. Because there is no mode switching between controlling the four variables of brightness, contrast, minimum input tone and maximum input tone, the interaction between these four variables can be easily and quickly explored. Experimental validation of this technique is not part of this thesis. However, it is highly likely that this technique will exhibit better performance than a single mouse technique, simply because there is one control rather than four.

The contributions of these techniques to the HCI community are twofold. First, they provide viable options for inclusion in graphics packages should a standard operating system start to support dual cursors. Second, they provide examples that designers can work from in creating symmetric interaction techniques for other tasks.

7.6 Subjective Evaluations

All four experiments included a post-experiment debriefing where subjects were asked to rank the conditions by preference. In all cases, the subjects were not told about their performance with each condition, so the responses were based only on the participant's perception of how he or she performed and how much he or she liked each technique. For the image registration experiment (simultaneous rotation, translation and scale), the splines experiment, and the TRC experiment, the participants strongly preferred the symmetric technique. However, for the simultaneous rotation and translation experiment, there was no clear preference amongst participants. This indicates that the debriefings were performed in an unbiased way, as the participants in the rotation and translation experiment did not report what I would have 'liked' them to report. The debriefing was done in the same fashion for all the experiments, so the positive subjective results for the other experiments can be trusted. Overall, these results support the hypothesis of the thesis: that users will prefer symmetric techniques for symmetric tasks.

For the TRC experiment, the subjective evaluation was much more detailed because of the use of the NASA TLX questionnaires (see Appendix C). It is regrettable that I was not familiar with these questionnaires earlier, as they provided a wealth of informative data regarding the relative mental and physical load placed on participants by the different techniques. The TLX

results from the TRC experiment also support the thesis hypothesis, because the symmetric technique has the lowest overall workload of the three techniques tested.

7.7 Summative Critique

While this thesis has shown the performance benefits of symmetric interaction techniques for symmetric tasks and has shown that users express preference for symmetric interaction techniques, there are issues with using symmetric interaction techniques. While symmetric interaction techniques obviously have a place in a UI designer's repertoire, the following issues should be considered.

System Integration The symmetric interaction techniques that were empirically tested for this thesis were tested in isolation. In other words, no testing was done of how symmetric interaction methods integrate with either asymmetric or single-mouse interaction methods. The symDraw application presented in Chapter 4 shows that this integration is possible. However, the effectiveness and user satisfaction of that integration has not been empirically evaluated. The positive results found in the empirical evaluations throughout this thesis should be considered for what they are: results that show the symmetric interaction techniques work well for isolated tasks. While I believe that the techniques work well within a larger (application-level) context, future work is needed to verify this.

Different Metrics The empirical evaluations presented in this thesis used time as the only metric. While time-based performance methods are important (and standard in HCI), they are not always the most salient. For some tasks, the time it takes may be inconsequential and the quality of the final results may be most important. Or there may be other metrics that make sense. UI designers should be aware of what metric is most suited for the task they are designing for.

Hand Requirements While standard desktop computers require the use of one hand for spatial input, the symmetric interaction techniques presented in this thesis require the use of two hands. In situations where a user only has one hand available to use for interacting with the interface (either because the user only has one hand or because one of the user's hands is needed for some other task), symmetric interaction techniques are obviously not a good choice.

Operating System and Window Manager Support Currently, most operating systems only support multiple spatial input streams at the lowest level. This includes Linux, MS Windows and Mac OSX. If you plug a second USB mouse a computer running any of these operating systems, the two mice will fight over control of the cursor. None of

these systems natively support dual-cursors. Thus, in order to run applications such as symDraw and symTone, the symPut driver must be used. This results in the user being presented with 3 cursors: one system cursor and two custom cursors used inside the applications. The system cursor must be used to interact with the desktop and has no effect inside the dual-cursor applications. The custom dual-cursors stay inside the applications and have no effect in the rest of the operating system. This issue will disappear when these systems adopt a more general spatial input paradigm.

Widget/Toolkit Support Because dual-cursor support is not standard, there are no high-level APIs for programming dual-cursor applications. This means that current user interface designers need to create custom widgets and tools for dual-cursor applications. This makes the creation of dual-cursor applications much more tedious and can lead to non-standard widget interaction. A standardized, dual-cursor widget library would make symmetric application development much more feasible.

7.8 Summary

This thesis has presented the Symmetric Interaction Model, which complements Guiard's Kinematic Chain Model for asymmetric interaction. The Symmetric Interaction Model will help GUI designers distinguish between tasks that are symmetric versus asymmetric so that an appropriate interaction style can be chosen for a given task.

This thesis has also presented a variety of symmetric interaction techniques and applications, both spatial and non-spatial. The symTone and symDraw applications are proof-of-concept applications that demonstrate a coherent presentation of symmetric interaction techniques combined with single-mouse and asymmetric interaction. The experimental results given in Chapters 4, 5 and 6 show that symmetric interaction techniques are more effective for naturally symmetric tasks than either single-mouse or dual-mouse asymmetric techniques. The symmetric interaction techniques for the tested tasks are also preferred by users over single-mouse and dual-mouse asymmetric techniques.

Symmetric interaction is an important component of two-handed interaction that has received little attention in the HCI literature. This thesis has attempted to fill that gap and provide a model which can be used by both designers and researchers. I have used standard USB mice for all of the implementations and experiments discussed in this thesis because I believe that two-handed interaction can be a standard desktop metaphor using inexpensive commodity devices. If a major operating system such as Windows or Macintosh begins to support multiple spatial input streams, I expect that the symmetric interaction model presented in this thesis will be used in conjunction with Guiard's Kinematic Chain model as important guidelines in the implementation two-handed desktop interaction.

Chapter 8

Future Work

As this thesis has contained a variety of technique implementations, the directions for future work are similarly varied.

8.1 Domain-Specific Research

The applications and techniques developed in this research have focused on 2-D graphics and image manipulation. There is scope for further symmetric interaction technique development in these areas, and there are many possibilities for symmetric interaction research in other domains. Some of these ideas and possibilities are described here.

8.1.1 Splines work

The symSpline technique demonstrates the expressiveness available when two hands can manipulate a deformable object such as a spline. An obvious extension to this research is to take the technique into three dimensions and apply it to surface manipulation, using two symmetric devices to manipulate a tangent plane. The degrees of freedom available from standard computer mice are not likely to be enough for this work – as with Llamas et al.'s Bender and Twister techniques [LKG+03, LPRS05], four degrees of freedom would be necessary. The mouse wheel could potentially be used for navigating or manipulating objects in a 3-D world, however, the 'jogs' on the wheel tend to make movement controlled by the mouse wheel bumpy rather than smooth, generating an interaction that is much less fluid. A good mapping from two standard mice to 3-D object manipulation is required and the best mapping is likely to be task-dependent.

8.1.2 Navigation

Tank interfaces have been studied in navigation and gaming contexts, usually using joysticks. This type of interaction is symmetric, but to my knowledge, has not been implemented using the scroll wheels on standard mice. The symPut driver would easily allow such interaction and could provide a very nice way to navigate 2-D spaces (in first-person view) using standard devices.

8.1.3 Gestural Manipulation

One of the most obvious ways that people use their two hands in real life is to assist with conversation by 'talking with their hands'. Hand gestures help to make communication more expressive. Hand gestures can also be used in human-computer interfaces. Research has been done on gesture recognition using data-gloves or sensors and cameras. However, it may be possible to do some simple gesture recognition with the symPut driver. For example, it should be relatively straightforward to recognize rotation, translation and scale gestures by the way the two hands are moving the mice. This could allow the user to simultaneously rotate, scale and translate objects without requiring the aspect ratio of the object to be constant.

8.1.4 Color Manipulation

The symTone application allows the user to symmetrically manipulate a tone-reproduction curve, as well as the input and output tone levels. Currently these manipulations are applied evenly to the red, green and blue color channels. It may be useful to allow users to manipulate the tone-reproduction curve separately for each color, or separately adjust the input and output tone levels for each color. However, there are likely to be much more interesting interactions that can be performed in manipulating 3-D color spaces. Another possibility is to use two mice to symmetrically select a color gradient or white balance.

8.1.5 Photograph Manipulation

Three of the techniques presented in this thesis have been aimed at manipulating digital images: the image registration technique involving rotation, scale and translation of images to create a panorama, the symTRC technique for tone-mapping, the ToneZone technique for tone-range manipulation and the symLens technique for comparing original and altered versions of a photograph. Another symmetric technique that could be applied to image manipulation is the specification of slide show pan and zoom transitions (commonly known as the Ken Burns effect) using a rectangle tool controlled by two cursors. A symmetric slide show application would nicely round out the 'syms' software suite.

8.1.6 Scientific Visualization

There are obvious applications of symmetric interaction techniques to 3-D scientific visualization problems. One possibility is using the two cursors to control the size and position of the base of a cone that has its tip centered in the middle of a 3-D data set. This would afford the user a unique and easily controlled view into the 3-D data.

8.1.7 Data Mining

The TRC experiment showed that symmetric interaction techniques can be used for manipulating non-spatial data. This affords the possibility of designing symmetric interaction techniques to allow fluid and expressive exploration of non-spatial information domains. Specifically, it seems likely that symmetric interaction techniques could be used to explore large, high-dimensional data sets.

8.2 Enabling Dual Cursors

The symPut driver described in Chapter 3 provides separate spatial input streams and has enabled the development of the symDraw and symTone applications. However, there are still two main obstacles that must be overcome before dual cursor interaction techniques can be easily introduced to the general population. The first obstacle relates to operating system and window manager support for dual cursors, and the second obstacle is in high level API support for developers.

8.2.1 The Dual Cursor Desktop

There are currently no major operating systems that support dual cursors on desktop, laptop or tablet computing environments. Applications such as symTone and symDraw use custom cursors in addition to the system cursor that is used for desktop operation. If window managers supported dual mice and dual cursors, the applications wouldn't have to draw custom cursors, and there could be two cursors instead of three. In this vein, I have developed a prototype dual cursor window manager application called Duo. Figure 8.2 shows a screenshot of duo in which the two cursors are being used for an operation involving the multiple selection of icons and the copying of those icons to a destination folder. By holding down the icons with one mouse and dragging them with another mouse, a copy operation is performed. Figure 8.2 shows the user stretching out a rotated window. Duo is not linked in to a real operating system, but it does provide a demonstration of what a dual-cursor window manager might look like. Further work on this application is necessary to demonstrate that a dual cursor window manager is useful. Specifically, future work in this area should show the following:

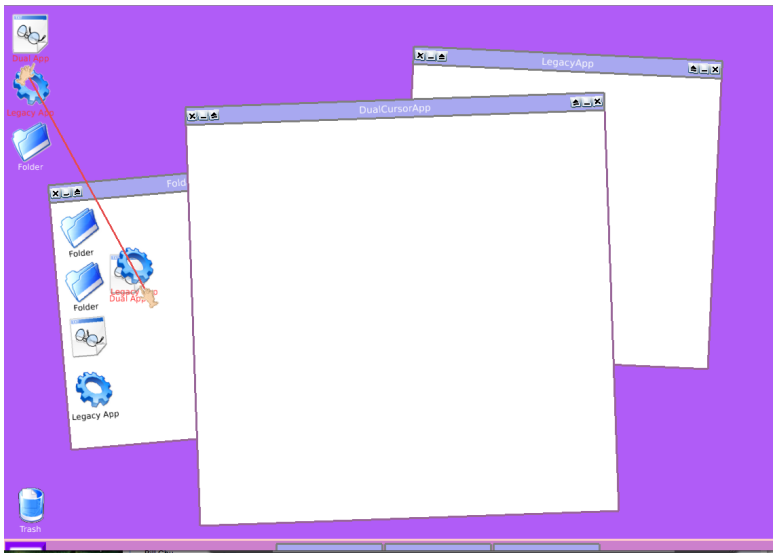


Figure 8.1: Using two cursors to select and copy multiple icons in Duo.

Figure 8.2: Using two cursors to stretch and position a rotated window in Duo.

- That having two cursors is not a hindrance or distraction to the user.
- That having two cursors enables new and useful interactions related to desktop organization.
- That having two cursors does not preclude the use of legacy single-cursor applications.
- That single cursor operations, asymmetric dual-cursor operations and symmetric dual-cursor operations can all coexist, allowing the most appropriate type of interaction for any given task.
- That a dual-cursor desktop presents an interface that is free of handedness bias.

8.2.2 Widget/Toolkit Support

The graphical user interface for the symDraw and symTone applications is custom built in order to support dual-cursor interaction. Although the symPut driver provides application designers with two separate spatial input streams, there is currently no support for dual-cursor interface widgets. Designing a full set of GUI widgets (such as buttons, menus, scroll-bars, drop-down lists, radio buttons and dialog boxes) that can handle dual-cursor interaction will not be a trivial undertaking. There are a variety of synchronization issues that will need to be addressed. The set of widgets will have to work in a coherent manner in order for the user to develop a mental model of what happens when the two cursors interact with one or more widgets at the same time. This is an area of research that is ripe with possibilities.

8.3 Advanced Form Factors

The work in this thesis was done using standard desktop and laptop computers and using standard USB mice as the devices. These form factors were chosen because it was desirable to show that two-handed spatial input is useful in typical computer settings, rather than in specialized settings with specialized equipment. However, there is tremendous opportunity for two-handed symmetric interaction in other form factors.

8.3.1 Large-Scale Displays

Smart Boards [Tec] and other large scale display devices already accept multiple inputs in some cases. This has been done to allow collaboration between multiple users. However, it also affords the ability for an individual user to interact using the two hands symmetrically. The widget toolkit support described previously would make this type of interaction easier to design.

8.3.2 Multi-Touch Interaction and Tablet Computing

Different types of multi-touch interaction technologies afford interesting possibilities for using both hands directly on a surface using either symmetrical or asymmetric interaction techniques. Recently, Han introduced the Multi-Touch Interaction technology that uses frustrated total internal reflection to sense multiple points on a surface [Han05]. This technology makes multi-touch interaction more practical. As this technology is developed and commercialized it will likely be used for tablet computing. This will afford the type of symmetric interaction described in this thesis. Because the technology can sense all 10 fingers, it will also afford much richer interaction, both symmetric and asymmetric. The Symmetric Interaction Model presented in this thesis should be a useful starting point for developing richer symmetric interactions for this technology.

8.4 The Psychology of Symmetric Interaction

While this thesis has presented a Symmetric Interaction Model and some applications and techniques that demonstrate the performance and usability of symmetric interaction for certain tasks, there is still more that could be discovered about the psychology of symmetric interaction.

8.4.1 Memory Cues

The ToneZone tool that is part of symTone allows users to quickly explore the interaction between input and output tone levels. As the user moves the rectangle around, he does not have to understand the input and output tone levels, but can merely find the size and position of the ToneZone rectangle that makes the underlying image look best, see Figure 8.3. One interesting aspect of ToneZone is that the user who wants to keep exploring the domain even after making the digital image look good can easily do so. He or she only needs to remember the size and position of the rectangle, rather than four individual control positions. Therefore, it seems likely that the ToneZone rectangle, while providing visual integration between the two cursors, also provides a spatial memory cue, which allows users to explore and then easily return to a configuration of interest. This should be investigated with a careful series of experiments to isolate the contributions of the ToneZone rectangle to both motor control and spatial memory.

8.4.2 Mental Model Development

Because the symmetric interaction techniques described in Chapters 5 and 6 allow novice users to easily explore a domain, it is possible that these techniques may be useful in helping novice

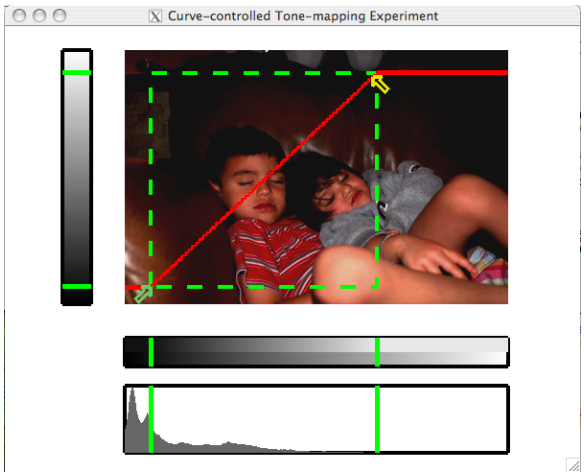


Figure 8.3: With symTone, users can use the position and size of the ToneZone rectangle as a memory cue that enables them to explore the domain and then return to a configuration that generated a good image.

users to develop a mental model of that domain more quickly than other techniques.

It would be very interesting to run an experiment with the symTRC technique and compare it to the standard Adobe Photoshop Curves tool. A good design might involve having experiment participants use one of the tools for two, five and ten minutes with a sample photo. At each time interval, the participant could be shown a new photo, and a separate tone-reproduction curve. The participant would then be asked to select from three or four final photos to choose how the photo would be changed by the given TRC. In this way, it would be possible to test how quickly the user develops a mental model of what the TRC is doing. A similar type of experiment could be conducted for novice splines users and for the ToneZone tool. If it was found that the symmetric techniques do allow users to develop mental models of complex domains more quickly, that would be a very important contribution to HCI.

8.4.3 Stimulus-Response Incompatibility

Another aspect of the psychology of two-handed interaction that requires more study is the impact of stimulus-response incompatibilities in computer interfaces. If two hands are used to manipulate a single object on screen, the act of rotating the object more than 90^o will cause the cursors to be on the 'wrong' side of the screen. It seems likely that the continuous feedback of the real-time interaction techniques tested in this thesis helped users to make sense of the flipped cursors. It also seems likely that the visual integration provided by the shapes being manipulated helped users to handle the stimulus-response incompatibility. However, both of

these hypotheses should be explicitly tested.

8.4.4 Tone-Matching

Three of the 21 participants recruited for the tone-matching experiment that was described in Chapter 6 did not finish the experiment. After an hour, these three participants were all still working on the initial practice trials. These three participants were all female, but it isn't clear from the data collected that gender is an issue (there was no statistically significant effect for gender in the ANOVA, but that was based on the 18 participants who finished the experiment). All three of these participants stated that the task was simply too hard for them. Why these three people found the task too hard, but the other 18 people were able to complete the experiment is unclear and warrants further investigation.

8.5 Future Work Summary

The Symmetric Interaction Model paves the way for future research in symmetric interaction techniques in a variety of domains and for a variety of form factors. More research can be done in examining the psychological impact of symmetric interaction in the areas of spatial memory cues for the user and the user's development of domain-specific mental models. The applications developed here demonstrate the necessity for the development of dual-cursor widget and toolkit support, as well as dual-cursor support at the window manager level.

Bibliography

[ABWW03] Alessandro Artusi, Jiři Bittner, Michael Wimmer, and Alexander Wilkie. Delivering interactivity to complex tone mapping operators. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 38–44, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[BB89] Richard Bartels and John Beatty. A technique for the direct manipulation of spline curves. In *Graphics Interface 1989 Proceedings*, 1989.

[BBA+00] Steve Benford, Benjamin B. Bederson, Karl-Petter Akesson, Victor Bayon, Allison Druin, Pär Hansson, Juan Pablo Hourcade, Rob Ingram, Helen Neale, Claire O'Malley, Kristian T. Simsarian, Danaë Stanton, Yngve Sundblad, and Gustav Taxén. Designing storytelling technologies to encouraging collaboration between young children. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 556–563, New York, NY, USA, 2000. ACM Press.

[BBB+93] Richard Bartels, John Beatty, Kellogg Booth, Eric Bosch, and Pierre Jolicoeur. Experimental comparison of splines using the shape-matching paradigm. *ACM Transactions on Graphics*, 12(3):179–208, 1993.

[BFKB99] Ravin Balakrishnan, George Fitzmaurice, Gordon Kurtenbach, and William Buxton. Digital tape drawing. In *UIST'99 Proceedings*, 1999.

[BFKS99] Ravin Balakrishnan, George Fitzmaurice, Gordon Kurtenbach, and Karan Singh. Exploring interactive curve and surface manipulation using a bend and twist sensitive input strip. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 111–118, New York, NY, USA, 1999. ACM Press.

[BH99] Ravin Balakrishnan and Ken Hinckley. The role of kinesthetic reference frames in two-handed input performance. In *UIST'99 Proceedings*, 1999.

[BH00] Ravin Balakrishnan and Ken Hinckley. Symmetric bimanual interaction. In *CHI 2000 Proceedings*, 2000.

[BKKK04] Seok-Hyung Bae, Takahiro Kobayashi, Ryugo Kijima, and Won-Sup Kim. Tangible NURBS-curve manipulation techniques using graspable handles on a large display. In *UIST 2004 Proceedings*, 2004.

[BL01] Michel Beaudouin-Lafon. Novel interaction techniques for overlapping windows. In *UIST 2001 Proceedings*, pages 153–154. ACM, 2001.

[BM86] W. Buxton and B. Myers. A study in two-handed input. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–326. ACM Press, 1986.

[BP98] Ravin Balakrishnan and Pranay Patel. The PadMouse: Facilitating selection and spatial positioning for the non-dominant hand. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 9–16, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[BS90] Teresa W. Bleser and John Sibert. Toto: A tool for selecting interaction techniques. In *UIST '90: Proceedings of the 3rd annual ACM SIGGRAPH symposium on User interface software and technology*, pages 135–142, New York, NY, USA, 1990. ACM Press.

[BSP+93] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: The see-through interface. In *Siggraph'93 Proceedings*, 1993.

[Bux86] William Buxton. Chunking and phrasing and the design of human-computer dialogues. In *Proceedings of the IFIP World Computer Congress*, pages 475–480, 1986.

[Bux87] W. Buxton. There's more to interaction than meets the eye: Some issues in manual. pages 366–375, 1987.

[Bux90] William Buxton. A three-state model of graphical input. In *INTERACT '90: Proceedings of the IFIP TC13 Third International Conference on Human-Computer Interaction*, pages 449–456. North-Holland, 1990.

[CFC03] Celine, Elodie Fourquet, and William Cowan. Two-handed colour selection. In *Graphics Interface '03 extended abstracts*, 2003.

[CGBL99] Didier Casalta, Yves Guiard, and Michel Beaudouin-Lafon. Evaluating two-handed input techniques: Rectangle editing and navigation. *ACM CHI'99 Conference (Extended Abstracts)*, pages 236–237, 1999.

[Cha94a] Stephane Chatty. Extending a graphical toolkit for two-handed interaction. In *UIST'94 Proceedings*, pages 195–204, 1994.

[Cha94b] Stephane Chatty. Issues and experience in designing two-handed interaction. In *CHI'94 Proceedings*, 1994.

[CK99] Chia-Yen Chen and Reinhard Klette. Image stitching - comparisons and new techniques. In *Lecture Notes in Computer Science*, volume 1689, pages 615–622. Springer-Verlag GmbH, Jan 1999.

[CMR90] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. The design space of input devices. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 117–124, New York, NY, USA, 1990. ACM Press.

[CMR91] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. A morphological analysis of the design space of input devices. *ACM Transactions on Information Systems*, 9(2):99–122, 1991.

[Com] PanaVue Company. Panavue image assembler. `http://www.panavue.com/`.

[CTS+97] Richard Carson, Julie Thomas, Jeffery J. Summers, Megan R. Walters, and Andras Semjen. The dynamics of bimanual circle drawing. *The Quarterly Journal of Experimental Psychology*, 50A(3):664–683, 1997.

[DCWP02] Kate Devlin, Alan Chalmers, Alexander Wilkie, and Werner Purgathofer. Star: Tone reproduction and physically based spectral rendering. In *State of the Art Reports, Eurographics 2002*, pages 101–123. The Eurographics Association, September 2002.

[DD94] Paul E. Dennison and Gail E. Dennison. *Brain Gym.* Edu-Kinesthetics, 1994.

[DIH+03] Jorn Diedrichsen, Richard B. Ivry, Eliot Hazeltine, Steven Kennerley, and Asher Cohen. Bimanual interference associated with the selection of target locations. *Journal of Experimental Psychology: Human Perception and Performance*, 29(1):64–77, 2003.

[ES01] David S. Ebert and Christopher D. Shaw. Minimally immersive flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 7(4):343–350, 2001.

[FB93] Barry Fowler and Richard Bartels. Constraint-based curve manipulation. *IEEE Computer Graphics and Applications*, 13(5):43–49, 1993.

[FD54] P.M. Fitts and R.L. Deininger. S-R compatibility: Correspondance among paired elements within stimulus and response codes. *Journal of Experimental Psychology: Human Perception and Performance*, 48:483–492, 1954.

[FIB95] George Fitzmaurice, Hiroshi Ishii, and William Buxton. Bricks: Laying the foundations for graspable user interfaces. In *CHI'95 Proceedings*, 1995.

[Fow90] Barry Fowler. Geometric techniques for interactive curve design. Master's thesis, University of Waterloo, 1990.

[Fow92] Barry Fowler. Geometric manipulation of tensor product surfaces. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 101–108, New York, NY, USA, 1992. ACM Press.

[FS94] Adam Finkelstein and David H. Salesin. Multiresolution curves. *Computer Graphics*, 28(Annual Conference Series):261–268, 1994.

[FWC84] J.D. Foley, V.L. Wallace, and P. Chan. The human factors of computer graphics interactive techniques. November 1984.

[FZSW01] Elizabeth A. Franz, Howard N. Zelaznik, Stephan Swinnen, and Charles Walter. Spatial conceptual influences on the coordination of bimanual actions: When a dual task becomes a single task. *Journal of Motor Behaviour*, 33(1):103–112, 2001.

[GBK+02] Tovi Grossman, Ravin Balakrishnan, Gordon Kurtenbach, George Fitzmaurice, Azam Khan, and Bill Buxton. Creating principal 3D curves with digital tape drawing. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 121–128, New York, NY, USA, 2002. ACM Press.

[GBS03] Tovi Grossman, Ravin Balakrishnan, and Karan Singh. An interface for creating and manipulating curves using a high degree-of-freedom curve input device. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 185–192, New York, NY, USA, 2003. ACM Press.

[GH97] S. Gibson and R. J. Hubbold. Perceptually-driven radiosity. *Computer Graphics Forum*, 16(2):129–141, 1997.

[Gor] Ryan C. Gordon. ManyMouse library. `http://icculus.org/manymouse/`.

[Gui87] Yves Guiard. Asymmetric division of labour in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behaviour*, pages 486–517, 1987.

[Han05] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05 Proceedings*, pages 115–118, New York, NY, USA, 2005. ACM Press.

[HB99] J.P. Hourcade and B.B. Bederson. Architecture and implementation of a java package for multiple input devices (MID). Technical Report HCIL-99-08, University of Maryland, 1999. CS-TR-4018 , UMIACS-TR-99-26.

[HCS98] Ken Hinckley, Mary Czerwinski, and Mike Sinclair. Interaction and modeling techniques for desktop two-handed input. In *UIST'98 Proceedings*, 1998.

[HIVB95] Beverly L. Harrison, Hiroshi Ishii, Kim J. Vicente, and William A.S. Buxton. Transparent layered user interfaces: An evaluation of a display design to enhance focused and divided attention. In *CHI'95 Proceedings*, 1995.

[HPP+97] Ken Hinckley, Randy Pausch, Dennis Proffitt, James Patten, and Neal Kassell. Cooperative bimanual action. In *CHI'97 Proceedings*, 1997.

[HPPK98] Ken Hinckley, Randy Pausch, Dennis Proffitt, and Neal F. Kassell. Two-handed virtual manipulation. *ACM Transactions of Computer-Human Interaction*, 5(3):260–302, 1998.

[HS88] S.G. Hart and L.E. Staveland. *Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research*, pages 139–183. North-Holland, 1988.

[IP05] Ali Iranli and Massoud Pedram. DTM: Dynamic tone mapping for backlight scaling. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 612–617, New York, NY, USA, 2005. ACM Press.

[JBFG93] Stanley Jang, Kellogg S. Booth, David R. Forsey, and Peter Graf. Investigating the effectiveness of direct manipulation of 3D B-spline curves using the shape-matching paradigm. In *Graphics Interface 1993 Proceedings*, 1993.

[JSMMPM94] Robert J. K. Jacob, Linda E. Sibert, Daniel C. McFarlane, and Jr. M. Preston Mullen. Integrality and separability of input devices. *ACM Trans. Comput.-Hum. Interact.*, 1(1):3–26, 1994.

[KBS94] Paul Kabbash, William Buxton, and Abigail Sellen. Two-handed input in a compound task. In *CHI'94 Proceedings*, 1994.

[KCST05] Russell Kruger, Sheelagh Carpendale, Stacey D. Scott, and Anthony Tang. Fluid integration of rotation and translation. In *CHI'05 Proceedings*, 2005.

[KFBB97] Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. The design of a GUI paradigm based on tablets, two-hands and transparency. In *CHI'97 Proceedings*, 1997.

[KMB93] Paul Kabbash, I. Scott MacKenzie, and William Buxton. Human performance using computer input devices in the preferred and non-preferred hands. In *InterCHI'93 Proceedings*, 1993.

[KSG79] J.A.S. Kelso, Dan L. Southard, and David Goodman. On the coordination of two-handed movements. *Journal of Experimental Psychology: Human Perception and Performance*, 5(2):229–238, 1979.

[LBCK06] Celine Latulipe, Ian Bell, Charles L.A. Clarke, and Craig S. Kaplan. symtone: Two-handed manipulation of tone reproduction curves. In *GI 2006 Proceedings*, 2006.

[LKC05a] Celine Latulipe, Craig S. Kaplan, and Charles L.A. Clarke. Simultaneous rotation and translation. In *Proceedings of HCI 2005, Volume 2*, 2005.

[LKC05b] Celine Latulipe, Craig S. Kaplan, and Charles L.A. Clarke. Unimanual and bimanual image alignment: An evaluation of mouse-based techniques. In *Proceedings of UIST 2005*, 2005.

[LKG+03] Ignacio Llamas, Byungmoon Kim, Joshua Gargus, Jarek Rossignac, and Chris D. Shaw. Twister: A space-warp operator for the two-handed editing of 3D shapes. *ACM Trans. Graph.*, 22(3):663–668, 2003.

[LMKC06] Celine Latulipe, Stephen Mann, Craig S. Kaplan, and Charles L.A. Clarke. symSpline: Symmetric two-handed spline manipulation. In *CHI 2006 Proceedings*, 2006.

[Loy] Jim Loy. Shuffling cards. `http://www.jimloy.com/math/shuffle.htm`.

[LPRS05] Ignacio Llamas, Alexander Powell, Jarek Rossignac, and Chris D. Shaw. Bender: a virtual ribbon for deforming 3d shapes in biomedical and styling applications. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 89–99, New York, NY, USA, 2005. ACM Press.

[LZB98] Andrea Leganchuk, Shumin Zhai, and William Buxton. Manual and cognitive benefits of two-handed input. *ACM Transactions on Computer-Human Interaction*, 5(4):326–359, 1998.

[MG01] I. Scott MacKenzie and Yves Guiard. The two-handed desktop interface: Are we there yet? In *CHI '01 extended abstracts on Human factors in computer systems*, pages 351–352. ACM Press, 2001.

[MKKP01] Franz Mechsner, Dirk Kerzel, Gunther Knoblich, and Wolfgang Prinz. Perceptual basis of bimanual coordination. *Nature*, 414:69–73, 2001.

[MMB93] Edgar Matias, I. Scott MacKenzie, and William Buxton. Half-qwerty: a one-handed keyboard facilitating skill transfer from QWERTY. In *Proceedings of the conference on Human factors in computing systems*, pages 88–94. Addison-Wesley Longman Publishing Co., Inc., 1993.

[MR97] Nobuyuki Matsushita and Jun Rekimoto. HoloWall: Designing a finger, hand, body, and object sensitive wall. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 209–210, New York, NY, USA, 1997. ACM Press.

[MW86] Merriam-Webster. *Webster's Ninth New Collegiate Dictionary*. Merrian-Webster, 1986.

[NU84] R. Nicoletti and C. Umilta. Right-left prevalence in spatial compatibility. *Perception and Psychopysics*, 35:333–345, 1984.

[ODSW04] Daniel L. Odell, Richard C. Davis, Andrew Smith, and Paul K. Wright. Toolglasses, marking menus, and hotkeys: A comparison of one and two-handed command selection techniques. In *Graphics Interface 2004 Proceedings*, pages 17–24. A.K. Peters, 2004.

[OKF+05] Russell Owen, Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. When it gets more difficult, use both hands - exploring bimanual curve manipulation. In *GI 2005 Proceedings*, 2005.

[Rai99] Roope Raisamo. An alternative way of drawing. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 175–182, New York, NY, USA, 1999. ACM Press.

[Rek02] Jun Rekimoto. SmartSkin: An infrastructure for freehand manipulation on interactive surfaces. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120, New York, NY, USA, 2002. ACM Press.

[RR96] Roope Raisamo and Kari-Jouko Raiha. Techniques for aligning objects in drawing programs. In *UIST'96 Proceedings*, 1996.

[RSSF02] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. In *SIGGRAPH '02*, pages 267–276, New York, NY, USA, 2002. ACM Press.

[SCB88] Maureen C. Stone, William B. Cowan, and John C. Beatty. Color gamut mapping and the printing of digital color images. *ACM Trans. Graph.*, 7(4):249–292, 1988.

[Ser] House Internet Services. PTgui. `http://www.ptgui.com/`.

[SG94] Chris Shaw and Mark Green. Two-handed polygonal surface design. In *UIST '94: Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 205–212, New York, NY, USA, 1994. ACM Press.

[SG97] Chris Shaw and Mark Green. THRED: A two-handed design system. *Multimedia Systems*, 5:126–139, 1997.

[Sin99] Karan Singh. Interactive curve design using digital french curves. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 23–30, New York, NY, USA, 1999. ACM Press.

[Sze04] Rick Szeliski. Image alignment and stitching. Technical Report MSR-TR-2004-92, Microsoft Research, 2004. `http://www.csie.ntu.edu.tw/~cyy/courses/vfx/papers/Szeliski2005IAS.pdf`.

[Tec] Smart Technologies. Smart boards. `http://www.smarttech.com/`.

[TG04] E. Tse and S. Greenberg. Rapidly prototyping single display groupware through the sdgtoolkit. In *CRPIT Conferences in Research and Practice in Information Technology Series*, volume 28, pages 101–110, 2004.

[Tse04] Ed Tse. The single display groupware toolkit. Master's thesis, The University of Calgary, 2004.

[WC90] Colin Ware and Bill Cowan. The RGYB color geometry. *ACM Transactions on Graphics*, 9(2):226–232, 1990.

[Wic00] Von Rainer K. Wick. *Teaching at the Bauhaus*. Hatje Cantz Publishers, 2000.

[ZFS97] Robert C. Zeleznik, Andrew S. Forsberg, and Paul S. Strauss. Two pointer input for 3D interaction. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 115–ff., New York, NY, USA, 1997. ACM Press.

[ZKSS99] Shumin Zhai, Eser Kandogan, Barton A. Smith, and Ted Selker. In search of the 'magic carpet': Design and experimentation of a bimanual 3D navigation interface. *Journal of Visual Languages and Computing*, 10:3–17, 1999.

Appendix A

Subjective Evaluation

The following comments were collected from participants at the end of the image registration experiment (see Chapter 4). The participants were asked if they had any comments about the experiment as a whole or about any of the particular techniques. Some participants had no comments. When the participant's comments refer to the 'first' or 'last' technique, the actual condition they are referring to (symmetric, asymmetric or single-mouse) is included in square brackets to clarify their comments.

Subject 1 It was fun, but it was pretty hard with one hand.

Subject 2 The one-handed one got frustrating and took longer.

Subject 3 It was easier when you could rotate and move at the same time.

Subject 4 The single mouse method was easier.

Subject 5 In the last one [asymmetric technique] you found youself adjusting the orientation of the left hand and it was difficult.

Subject 6 That was funner than Pscyh experiments.

Subject 8 The asymmetric one was more weird and difficult.

Subject 12 The pictures got boring after awhile.

Subject 11 The single mouse switching between moving and rotating was hard.

Subject 16 It was tiring on the eyes.

Subject 15 The harder texture (the one with the big oval and the two lines inside it) was REALLY hard with the one-handed technique.

Subject 18 I was surprised at how easy the two mice symmetric technique was.

Subject 17 It was weird at first using two mice, but then it was easy.

Subject 19 The symmetric technique was best.

Subject 20 It was a lot easier with two mice.

Subject 21 It was definitely easier with two mice.

Subject 23 The first method [symmetric] was easiest. The other 2 got irritating.

Subject 24 I thought the two two-handed techniques were the same.

Appendix B

Experiment Letters

All experiments described in this thesis had the approval of the University of Waterloo Office of Research Ethics. The Office of Research Ethics has strict standards regarding the paperwork that must be kept when doing research with human subjects. The following letters are examples of the letters that were given to participants during the four experiments described in this thesis. The letters below are from the splines experiment, the letters for the other three experiments are very similar. The letters were printed on University letterhead.

The following three pages present the Information Letter, the Consent Form and the Thank-you Letter.

B.1 Splines Experiment Information Letter

January, 2005

Title of Project: Human Capabilities in Using Two Hands, Two Mice, Two Cursors

Student Investigator: Celine Latulipe

Faculty Investigators: Charlie Clarke and Craig Kaplan, University of Waterloo, School of Computer Science, (519) 884-4567 Ext. 4548

You are invited to participate in a study that concerns how people can use two hands to control two cursors on a computer screen with two computer mice. As a participant in this study, you will be asked to complete a series of tasks involving manipulating curves on a computer screen using your right hand and using both hands together. You will be asked to match two curves as quickly as possible.

Participation in this study is voluntary, and involves a single session which will take less than one hour of your time. You can stop participating at any time by notifying the researcher. By volunteering for this study, you will learn about two-handed computer interfaces. In addition, you will receive $10 for your participation at the end of the session. You may decide to withdraw from this study at any time by advising the researcher, and may do so without any penalty. All information you provide is considered completely confidential; indeed, your name will not be included or in any other way associated, with the data collected in the study. Furthermore, because the interest of this study is in the average responses of the entire group of participants, you will not be identified individually in any way in any written reports of this research. Data collected during this study will be retained indefinitely, on a secure computer account to which only researchers associated with this study have access. There are no known or anticipated risks associated to participation in this study.

I would like to assure you that this study has been reviewed and received ethics clearance through the Office of Research Ethics at the University of Waterloo. However, the final decision about participation is yours. If you have any comments or concerns resulting from your participation in this study, please contact Dr. Susan Sykes at this office at (519) 888-4567 Ext. 6005.

Thank you for your assistance in this project.

B.2 Splines Experiment Consent Form

I agree to participate in a study being conducted by Celine Latulipe of the School of Computer Science, University of Waterloo. I have made this decision based on the information I have read in the Information-Consent Letter and have had the opportunity to receive any additional details I wanted about the study. I understand that I may withdraw this consent at any time by telling the researcher without penalty.

I also understand that this project has been reviewed by, and received ethics clearance through, the Office of Research Ethics at the University of Waterloo, and that I may contact this office if I have any concerns or comments resulting from my involvement in the study.

Name (print):

Signature:

Date:

Witness Signature:

B.3 Spline Experiment Thank-you Letter

January, 2005

Dear Participant,

I would like to thank you for your participation in this study. As a reminder, the purpose of this study is to identify the ability of humans to use two hands to control two cursors on a computer screen using two computer mice.

The data collected during the study experiments will contribute to a better understanding of the usefulness of two-handed computer input and will help in the design of two-handed computer interfaces.

Please remember that any data pertaining to you as an individual participant will be kept confidential. Once all the data are collected and analyzed for this study, I plan on sharing this information with the research community through seminars, conferences, presentations, and journal articles. If you are interested in receiving more information regarding the results of this study, or if you have any questions or concerns, please contact me at either the phone number or email address listed at the bottom of the page. If you would like a summary of the results, please let me know now by providing me with your email address. When the study is completed in February 2005, I will send it to you.

As with all University of Waterloo projects involving human participants, this project was reviewed by, and received ethics clearance through, the Office of Research Ethics at the University of Waterloo. Should you have any comments or concerns resulting from your participation in this study, please contact Dr. Susan Sykes in the Office of Research Ethics at 519-888-4567, Ext., 6005.

Celine Latulipe

University of Waterloo School of Computer Science

(519) 888-4567 ext.4548 clatulip@cgl.uwaterloo.ca

Appendix C

The NASA Task Load Index Questionnaire

C.1 Introduction

The NASA Task Load Index (TLX) questionnaire was used in the tone-reproduction curve experiment that is described in Chapter 6. This questionnaire is a standard form used in the HCI community to assess the difficulty of a given task. In the TRC experiment, each participant performed 20 tone-matching trials for each of three conditions: symmetric bimanual TRC manipulation, asymmetric bimanual TRC manipulation and single-mouse TRC manipulation. The order of condition presentation was counter-balanced across the participants. Each participant filled out the NASA TLX questionnaire after finishing the trials for each of the three conditions.

The NASA TLX questionnaire is two pages. The first page requires the respondent to rate six categories on a scale of 0 to 20. The second page requires the respondent to rank pairs of the categories by which was more important to them in completing the task. The questionnaire is presented on the following two pages.

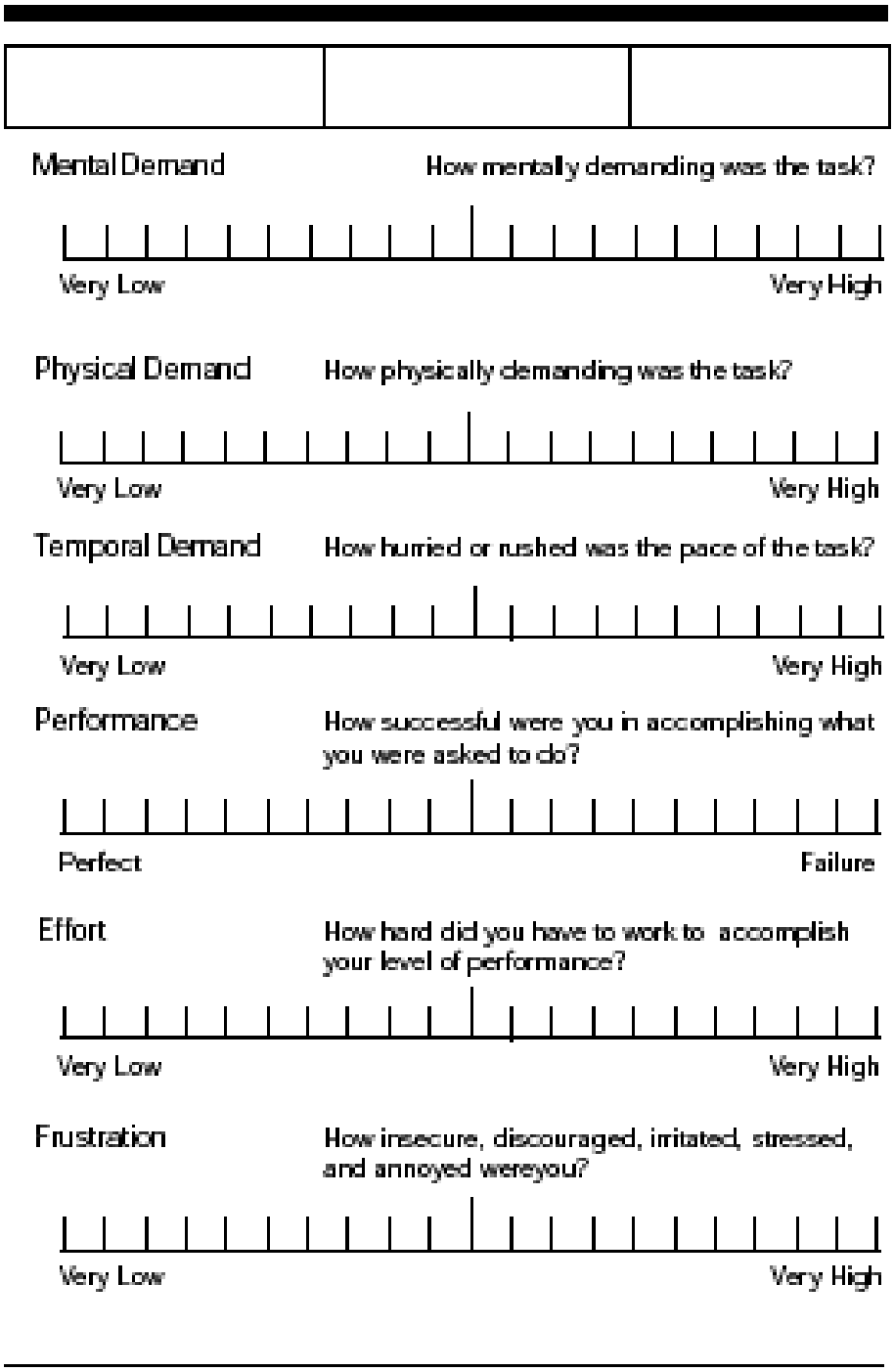


Figure C.1: NASA TLX questionnaire, first page.

Task Demand Comparisons

For each of the following, please place a mark to indicate which of the two was more demanding for you.

1. _____	Physical Demand	OR	_____Mental Demand
2. _____	Temporal Demand	OR	_____Mental Demand
3. _____	Performance	OR	_____Mental Demand
4. _____	Frustration level	OR	_____Mental Demand
5. _____	Effort	OR	_____Mental Demand
6. _____	Temporal Demand	OR	_____Physical Demand
7. _____	Performance	OR	_____Physical Demand
8. _____	Frustration level	OR	_____Physical Demand
9. _____	Effort	OR	_____Physical Demand
10. _____	Temporal Demand	OR	_____Performance
11. _____	Temporal Demand	OR	_____Frustration level
12. _____	Temporal Demand	OR	_____Effort
13. _____	Performance	OR	_____Frustration level
14. _____	Performance	OR	_____Effort
15. _____	Effort	OR	_____Frustration level

Figure C.2: NASA TLX questionnaire, second page.