

A Camera-Based Approach to Remote Pointing Interactions in the Classroom

by

Francisco Escalona Gonzalez

B.Sc., Universidad Nacional Autónoma de México, 2008

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

September 2015

© Francisco Escalona Gonzalez 2015

Abstract

Modern classrooms are brimming with technological resources, from sound systems and multiple large wall displays to the individual computers and cellphones of students. Leveraging this technology presents interesting opportunities in human-computer interaction to potentially create a richer experience that improves the effectiveness of classrooms both for lecturers and students. A key enabler of this experience is the ability for lecturers to manipulate objects on the screen by direct pointing, allowing them to be away from a computer for input. We designed and implemented a remote pointing technique that makes use of a web-camera and a pattern of shapes on the wall display to perform target tracking. A controlled study to evaluate performance and compare the camera-based technique to a traditional mouse for a target selection task in a classroom setting revealed that both devices have comparable error rates but that users are almost twice as fast with the mouse. The increased freedom of movement and immediacy of interaction provided by direct pointing makes the trade-off between speed and convenience reasonable. The technique does not require specialized hardware: the ubiquity of personal pocket cameras and computers makes target tracking with a camera a feasible future option for enabling direct pointing interactions on large wall displays in classroom settings.

Preface

The research presented in this thesis was carried out under the supervision of Dr. Kellogg S. Booth. I was the primary researcher in all work presented. Peter Beshai provided the code for the i>Clicker driver that was used in the implementation of our pointing device.

Ethics approval for the experimental study with human participants was provided by the Behavioural Research Ethics Board at UBC under ID H11-01756.

The research reported in this thesis was funded under the Discovery Grant program by the Natural Sciences and Engineering Research Council of Canada and under the Network of Centres of Excellence program through GRAND, the Graphics, Animation and New Media Network of Centres of Excellence.

Table of Contents

Abstract	ii
Preface	iii
Table of Contents	iv
List of Tables	vii
List of Figures	viii
Acknowledgements	xi
Dedication	xii
1 Introduction	1
1.1 Two Important Input Devices for Pointing	4
1.1.1 Light Pen	6
1.1.2 Mouse	10
1.2 Contributions	15
1.3 Overview of the Thesis	16
2 Related Work	17
3 Chiroptic Tracker: Camera-Based Remote Pointing	24
3.1 Design Constraints	24
3.2 The Camera Tracking Process	26
3.3 Implementation	32
3.3.1 Grid of Markers	33
3.3.2 Cursor	35
3.3.3 Feature Extraction	36
3.3.4 Computing Relative Coordinates	37

Table of Contents

3.3.5	Cursor Position	40
3.3.6	Performance	42
3.4	Known Limitations	42
3.4.1	Occlusion Caused by the Grid	42
3.4.2	High Color Contrast	44
3.4.3	Chaotic Movement	45
3.4.4	Lens Blur	45
3.4.5	Lag	45
3.4.6	Acute Angles	46
3.5	Pilot Testing	47
4	Comparing Remote Pointing to the Mouse	50
4.1	Hypotheses	50
4.2	Empirical Models of Pointing Performance	51
4.3	Method	54
4.3.1	Participants	54
4.3.2	Apparatus	55
4.3.3	Study Design	59
4.3.4	Procedure	60
4.4	Results	62
4.4.1	Movement Time	63
4.4.2	Error Rates	65
4.4.3	Throughput	66
4.4.4	Subjective Data	68
4.5	Discussion	70
5	Design Brief for a Classroom Interface	72
5.1	Goals of the Interface	72
5.2	Required Resources Available Today	74
5.3	Styles of Interaction	75
5.4	General Recommendations	76
5.5	Example of Future In-Classroom Interaction	77
6	Conclusions and Future Work	80

Table of Contents

Bibliography	83
-------------------------------	----

Appendix

A Experimental Resources	90
A.1 Consent Form	90
A.2 Initial Questionnaire	93
A.3 Final Questionnaire	95

List of Tables

4.1	The six <i>pose</i> conditions used in the study.	60
4.2	The four formulations of Fitts's Law considered for our analysis. Each predicts movement time MT from target width W and amplitude (distance) of movement A	64
4.3	Movement time models for the Fitts and Welford formulations. Significant differences in nested models are highlighted in bold.	64
4.4	Movement time models for the Shannon-Fitts and Shannon-Welford formulations. Significant differences in nested models are highlighted in bold.	64
4.5	Error rates for <i>pose</i> and <i>target</i> conditions.	65
4.6	Throughput means for all <i>pose</i> conditions, in bit/s.	66

List of Figures

1.1	The difference between relative and absolute pointing. In relative pointing (left) the cursor's final position P_f depends on both the movement of the device and the cursor's initial position, P_i . In absolute pointing (right) the cursor's final position depends only on the device's final position.	5
1.2	The light pen (top) and a diagram showing its composition (bottom). Taken from Sutherland's dissertation.	8
1.3	A replica of the original mouse by Engelbart and English seen from its side and bottom. Images borrowed from the Computer History Museum's webpage.	11
1.4	Optical system for a mouse. A light source, in this case an LED, shines through a plastic lens to illuminate the surface, while a small chip that contains a tiny camera processes the images to detect translation changes from one frame to the next. Image by Jeroen Domburg from spritesmods.com. . . .	13
3.1	The three processes involved in chiroptic tracking. The display renders the cursor and screen contents, the sensor interprets what it sees in the display to extract a position, and the human brain adjusts for errors.	27
3.2	Two different perspectives of the same grid cell, as captured by the camera. Corresponding points are labeled with the same letter. The tracker has to deal with the distortions caused by perception. Note also the marked contrast in illuminations on the top and bottom sides of the left image. . . .	29
3.3	The architecture of our tracker's implementation.	32

List of Figures

3.4	Grid of fiducial markers. The black ellipses define a reference frame, the gray circles determine proper orientation, and the green squares encode row and column number.	33
3.5	A sequence of bits used to encode row and column numbers. Numbers from 0 to 15 are arranged so that the two high-order bits of one are the same as the two low-order bits of the next. Each of the 16 numbers appears once in the sequence.	34
3.6	Design of the cursor for the chiroptic tracker.	35
3.7	The results of feature extraction on one frame of the camera. On the left is the original image with perimeter pixels and the main axis of each ellipse highlighted in red. On the right is the the binarized version used to find the ellipses, and their bounding boxes.	37
3.8	(Top) Difference between affine (a) and perspective (b) transformations. (Bottom) Although the affine transformation based on points A , B and C is not good enough to predict the position of point D , it finds a good approximation in point D'	39
3.9	An early version of the chiroptic sensor that incorrectly identified four blobs, highlighted with red, green, blue and magenta, as the corners of a grid cell. The cause is extreme perspective distortion in the image and the use of weaker heuristics for grid identification.	46
3.10	Two of the grips used to hold the chiroptic tracker. The <i>screwdriver grip</i> on the left is commonly used to hold a laser pointer. The <i>pencil grip</i> on the right seems to be slightly more intuitive for aiming the tracker.	49
4.1	Illustration of the task performed by participants: (left) the cursor is moved from its starting position towards the darker rectangle, (center) if the user clicks correctly on the target the other rectangle becomes the new target, and (right) if instead the user misses the target it flashes red and the targets then switch.	55

List of Figures

4.2	(top) A picture of the room illustrating what participants saw. All lights were turned off during the study. (bottom) A diagram drawn to scale of the room layout. Positions #1 and #2 are perpendicular to the center of the screen, and position #3 is off to the side.	56
4.3	The physical prototype of the tracker device used by participants. The image also shows the pencil grip they were asked to use during the study.	59
4.4	Mean movement times in milliseconds at different indexes of difficulty for all <i>pose</i> conditions. Lines are included only for readability.	63
4.5	Comparison of average throughput values per participant. Participants were sorted by increasing throughput in the “no grid” condition, and the values for the “standing” condition are also shown.	67
4.6	Difficulty ratings subjectively reported by participants for the pointing task using the chiroptic tracker, from 1 (easy) to 5 (impossible).	68

Acknowledgements

Time flies when you are having fun, and two years at UBC can go by very quickly. Still, much has happened during this time that has allowed me to grow as a person, and I owe a debt of gratitude to many people who made my experience unforgettable.

First and foremost, I am thankful to all the people who took the time to share their knowledge with me and challenge me; my teachers, you are outstanding. Among them I have to single out my supervisor, Dr. Kellogg Booth, whose mind is a cornucopia of interesting ideas and projects, and whose kindness has given me space to grow in my field. Thank you for your time and your conversation.

One of the salient features that I love about the CS department at UBC is the willingness of its people to help each other, more so when everyone's time is so constrained. I am thankful to my second reader, Dr. Jim Little, for his advice on this work that has made it better, and to all the others who likewise provided guidance and insight on my work.

My friends and fellow grad students are among the most capable and smart people I have met; working with them has been a pleasure. Thank you, Antoine, Kamyar and Derek, for our work together, and all the people in the MUX lab who inspire me with their work and friendship, especially Peter, Matt, Jessica and Oliver, for advice on things academic and otherwise.

To my family, most of them encouraging me from afar, but always present. And to my wife, Jazmin, for the past and for the future.

I am blessed.

*To Lulu, Héctor and Edel,
for everything.*

Chapter 1

Introduction

The chiroptic¹ tracker is a pointing input device designed for instructors as a means of interacting directly with content on classroom displays. Those displays are usually large and out of reach, so the interaction happens at a distance and from all kinds of angles. The chiroptic tracker technique is based on the metaphor of pointing a camera at the large wall display as if a line were shooting out of its center to interact with the display at the point of intersection. We achieve this direct pointing interaction by interpreting the contents of the image captured by the camera, and figuring out a position on the screen to render the cursor. The tracker is aimed in a similar way as a laser pointer, but does not require more hardware resources than those currently available to instructors.

Modern classrooms are equipped with lots of resources: multiple displays driven by powerful projectors, sound systems, computers, networking, lighting controls, and all the personal devices contributed by its occupants, which means more displays, microphones, processors, cameras, etc. These resources are not being used to their full potential. Imagine some future time when all of these devices are the instruments of an orchestra: they tune and synchronize at the beginning of the class, identifying each other and their layout around the room, so that as the class develops they create a richer experience than what they could each bring in isolation. When

¹ This is a word we made up from two roots, *chiro-* and *optic*, defined in the Collins English Dictionary as:

chiro- *combining form* indicating the hand; of or by means of the hand
optic *adj.* (Anatomy) of or relating to the eye or vision

We are aware of a previous meaning of the word used in Chemistry to describe optical techniques for investigating chiral substances, but that is not how we will use the word. We use it to qualify an object that allows a person to “see through their hands”, possibly by holding a camera.

students ask a question, microphones around them pick it up and send it to speakers at the other end of the room to echo it so that everybody can hear. Each personal laptop, tablet, or smartphone screen is a window into the content shown on the main displays that students can annotate together. The lecturer can move around freely and have conversations that engage everyone, and when an unexpected question comes up, both students and the lecturer have the freedom to modify the display without having to go all the way back to the lectern. These are real possibilities that can be explored with what is available right now. Direct pointing is an enabling capability for this kind of experience.

The highly specialized classrooms for subjects such as Geography, Chemistry and Art, with posters and tools always at hand, can be seen as interfaces that provide a better user experience. These classrooms make learning subject matter easier because the maps, burners and vials, or art supplies the class needs at any given time are readily accessible. A Geography classroom that does not have a map of a territory posted on a wall for reference when a class is trying to learn about its terrain is not doing its job well. In Don Norman's terminology [34], it does not have good affordances for learning Geography. Other subjects such Literature or History might require fewer visual displays, but they should present an interface conducive for discussion. Mathematics might require many surfaces to write on. In sharp contrast to these examples, classrooms that are multi-purpose like those commonly found in universities are stripped down of any specific content, and instead provide tools so that lecturers can "dress" them however they see fit for their class, which often means the lecturer prepares beforehand a set of slides to show on the room's main displays, to a great extent establishing the way the lecture will flow over time. This approach to learning is less organic and less flexible, and often lacks engagement for students.

Consider what a lecturer that wants to improvise would have to do to modify the contents of the screen. They often only have a limited mechanism that allows them to move back and forth between slides, so for a more complex maneuver they have to physically approach their computer to use the keyboard and mouse, getting engaged in their *personal* screen, setting

things up and then glancing back towards the display to check that it is showing what they need before resuming where they left off. If we acknowledge that lecturing is a little like putting on a show, then this interaction is terrible because it leaves the audience unattended, completely ignored by the performer, disrupting their attention and the flow of the conversation. The interface is in the way of the users. These limitations of the classroom can be overcome with better interfaces, and direct pointing is a crucial element.

It is not that current input devices do not work, but they were designed for a different kind of environment, a more personal one. If we use them naively in the classroom, they create an obstacle between the users and what they want to do. Current devices do not offer good affordances for the kind of interaction we believe is important in the modern classroom.

In this thesis we explore the problem of direct pointing on large wall displays in a classroom setting, using technology already available to lecturers and without prior calibration to the specific classroom in which it is deployed. We design a technique that matches the requirements that we identify, we describe its implementation, then we test it against the mouse in a pointing task.

Our focus is on providing an improved input technique *for the lecturer*. While members of the audience could also use the technique presented here, they already have better means of interacting with the contents of the display, such as their network-connected personal computers, or personal response systems such as the i>Clicker. Previous research by Beshai [6] and Shi [43] has already looked into that side of classroom interactions, so we will not go further into it here. For an example of a system that allows the audience to share and control the large display, Liu and MacKenzie offer LA-COME [23, 28] as a solution. In any case, it is worth noting that improving the experience for the lecturer will likely also improve the experience for the audience, by creating a more engaging and dynamic classroom environment.

Two fundamental assumptions guide our research:

1. Remote pointing in the classroom can be achieved without introducing any more hardware resources than those commonly available to

lecturers: a general-purpose computer controls the contents of a large wall display, and the lecturer holds a camera that can communicate with the computer. Nothing else is required.

2. The lecturer's interactions with the display are sporadic. Among other things, this implies that a trade-off between speed for the convenience of the lecturer and clarity for the engagement of the audience is acceptable.

1.1 Two Important Input Devices for Pointing

Pointing devices can be used for either *relative* or *absolute* input, and some can be used for both modalities. Absolute pointing means that there is a 1–1 correspondence of positions in physical space to positions in the virtual display, so that physically pointing or moving to the same physical position will always give the same result in virtual space. A prime example of this is a touch-sensitive surface, such as modern tablets and smartphones, where touching a part of the screen means also “clicking” on whatever is shown at that position in the display. In contrast, other devices have relative positioning, like the mouse, where the cursor's final position depends on its initial position on the screen, even when the exact same movement is performed with the mouse on a surface. Figure 1.1 illustrates these concepts.

Another aspect of pointing devices is that their *control to display ratio* (or its reciprocal, *gain*) can be adjusted to tune their sensibility. Gain is a multiplicative factor that is applied to the movement of the device to increase or decrease the movement in the virtual space. A small movement in a mouse with high gain can make the cursor go from one side of the screen to the other. This has the advantage that users have to spend less effort to move bigger distances in virtual space, but comes with the trade-off of lowering their precision, because it is harder to point at things precisely when even the slightest movement will cause a large change in position. On the other hand, a low gain value means that the device has to be moved a large distance to make the cursor go from side to side.

Users of the mouse who have experienced low gain settings are probably

1.1. Two Important Input Devices for Pointing

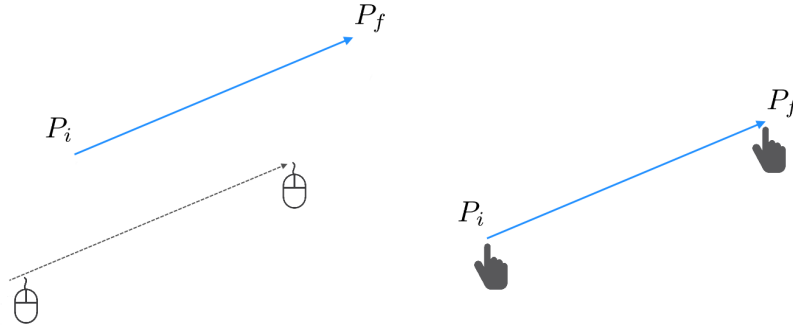


Figure 1.1: The difference between relative and absolute pointing. In relative pointing (left) the cursor’s final position P_f depends on both the movement of the device and the cursor’s initial position, P_i . In absolute pointing (right) the cursor’s final position depends only on the device’s final position.

familiar with a technique in which, after moving the mouse through a large displacement, it is raised and brought back and moved again, repeating this perhaps many times to reach the desired target destination. When the mouse is raised it “stops working”, in the sense that it does not change the position of the cursor on the screen while it moves through air, but as soon as it is set back down the cursor starts moving again. This technique is called *clutching* and can be used in other devices as well to temporarily disable them while the user gets into position, preventing unwanted movement of the cursor. Naturally, an interaction that requires much clutching takes more time and can be frustrating, but low gain values increase precision.

Modern operating systems can choose to vary the control to display ratio of a pointing device, and often do this dynamically based on the speed of movement as the user moves around, often without them noticing. When they detect a fast movement they increase gain, making the assumption that the user wants to go to a far target, but the value is decreased for smaller movements to improve accuracy and precision. Varying gain dynamically can improve user’s performance with a device, automatically balancing speed and precision trade-offs.

Gain is most often associated with relative pointing, but it can also be

1.1. Two Important Input Devices for Pointing

used for absolute pointing. The defining property of absolute pointing is that there is a 1–1 correspondence between control position (the location of the input device) and display position (the location of the cursor). When the correspondence has no scale factor, the gain is one, but when there is a scale factor between control and display the correspondence is still 1–1.

Because both relative and absolute pointing mechanisms can make use of clutching and gain, in a remote pointing setting like the one we are interested in clutching can be used to activate the device only when the user needs it, and get it out of the way when they do not. This blurs the distinction between relative and absolute pointing, the difference being more about how clutching is triggered than about whether the mapping is relative vs. absolute.

Gain occurs naturally in absolute pointing: if users are close to the screen, the angles involved create a low gain that increases precision, and conversely when users are far away the angles create a high gain that allows them to reach all corners with reduced effort. Gain can be provided in software to simulate this, somewhat analogous to how camera angle changes viewing perspective. Changing gain for absolute pointing changes the angular mapping for cursor location.

In the next sections we briefly look into other devices that are relevant to our research for a variety of reasons: their popularity, the ideas they bring to the discussion, or because they are used in settings similar to the classroom where there is a large display and an imbalance of roles between lecturer and audience. We describe how each device works and highlight key ideas, weaknesses or strengths that they exhibit, with the purpose of informing our own design. Most of the historical claims in this Section are based (without explicit citation) on Buxton’s excellent historical survey [8]. Other sources are cited explicitly in context.

1.1.1 Light Pen

In the early 1950’s Robert Everett created an input device called the Light Gun that looked like an actual gun with a handle and barrel, which the

1.1. Two Important Input Devices for Pointing

user pointed to the screen to read the position of an object by pressing the trigger. The Light Pen was descended from it, designed in 1957 by Ben Gurley to allow users to interact with a computer via a stylus-shaped input device pointed directly at the screen. It consisted of a light sensor tuned to detect the distinct light signal emitted by the phosphor-coated glass in CRT displays of the era. The electron beam of the display would activate the phosphor at specific positions one dot at a time, and the light pen reacted when it saw the initial peak of the light emission. By keeping a record of the position of the dots being rendered, the pen's reaction told the computer roughly at what part of the screen the user was aiming, enabling direct pointing input on the display.

Ivan Sutherland introduced Sketchpad [46] to the scientific community in 1963, a system that implements drawing on computer displays directly using a light pen as a pointing device. In his PhD dissertation he describes many ideas for interacting with computers in graphical ways. Some of the concepts he discusses have been adopted in modern interfaces, including the notion of direct manipulation by pressing a button to select an object or drag it around. A light pen like the one shown in Figure 1.2 was a central piece in his work, because it enabled a type of interaction with the computer that people could relate to well — it was like drawing on the screen — giving the user a greater freedom of movement and expression that was not possible using only buttons and knobs. Sutherland described in detail the techniques he designed to track the light pen's position across the screen continuously, and some finer details of his implementation that serve as inspiration when designing other pointing devices.

A critical point is that the sensor needed to see some light in order to know its position, but when the user moved it away from the light source the position was lost again. In order to track the pen continuously the system needed to make sure there was always some shape being drawn under it, or close enough that it could be seen. A clever solution to this problem was the use of the cursor. The cursor is a visual indicator of what position the user is pointing at, but it is itself a shape on the display, and if it is moved fast enough mirroring the light pen there will always be a shape to see and thus

1.1. Two Important Input Devices for Pointing

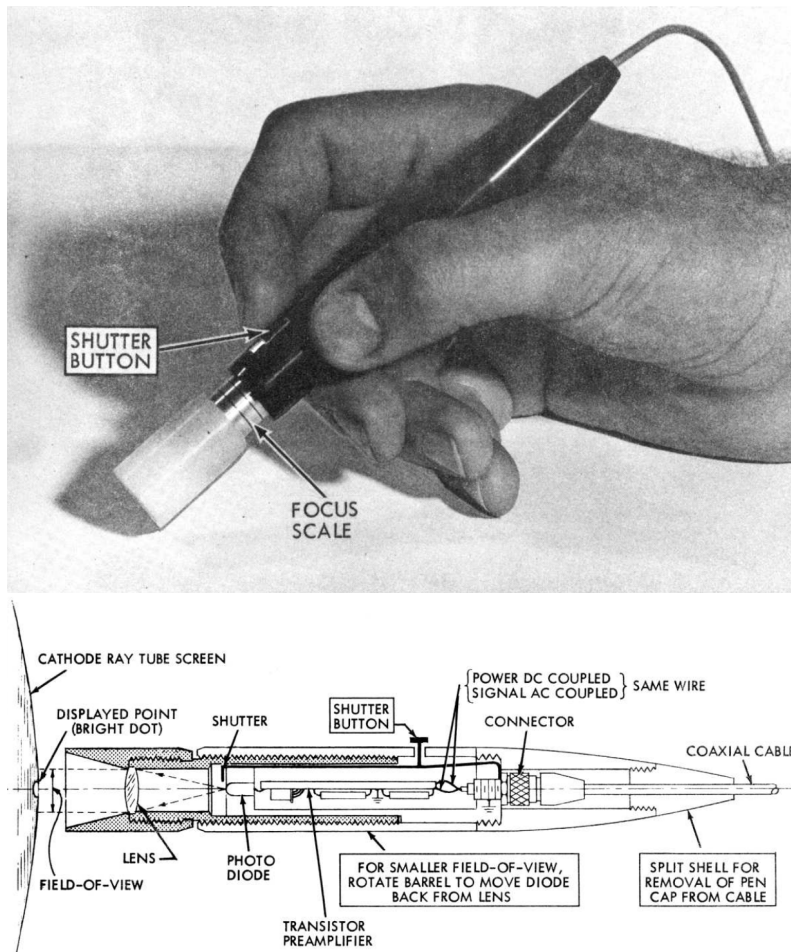


Figure 1.2: The light pen (top) and a diagram showing its composition (bottom). Taken from Sutherland's dissertation.

to get position information from, creating a nice loop that enables tracking. With a field of view of about half an inch on the light pen, Sutherland was able to follow the path of the user's movement at a speed of 20 inches per second across the screen if he refreshed the cursor position 100 times each second. This tracking process alone took 10% of the computer's resources on the first version Sutherland implemented, but it worked well enough that he could develop his interface, Sketchpad.

1.1. Two Important Input Devices for Pointing

Users would first have to activate the light pen by pointing it at any shape on the screen, a process Sutherland called “inking up”, which would cause a cursor to be drawn to indicate that the light pen was now tracking. They would then move it to perform whatever action was required, and when they were done they would just have to flick it fast enough to make the sensor lose track of the cursor, causing it to disengage. That was the clutch mechanism for the light pen. A user would have to learn how to perform these three actions: acquiring the cursor, tracking, and releasing the cursor, to use the light pen effectively in his system.

Different cursor designs were tried for the light pen, from a cloud of random points to a tracking cross built from individual dots that could be spaced apart evenly, or as Sutherland chose to do it, logarithmically, presumably because this increases the concentration of points at the edges of the cross, which is the portion most likely to be in the sensor’s field of vision. He also experimented with the idea of anticipating cursor movement by using constant velocity and constant acceleration equations, although he reports that those techniques caused instability and he did not pursue them further. In Chapter 3 we will talk about why this happens, and what can be done to deal with it.

Sutherland gives another important insight in making a distinction between “actual” and “pseudo” locations. He realized that the actual position the user is pointing to is often not as important as what they *mean* to point to, which in the case of Sketchpad could be a line, a point of intersection, or even an abstract quantity such as the length of a line segment. He used a host of geometry heuristics and threshold values to try to tease out this information from the user’s actions, and would often render the cursor at its pseudo position to give better visual feedback to the light pen holder.

This is an important lesson of the light pen that we want to stress: the cursor is an efficient design that has multiple purposes, directed to different observers. On the one hand it indicates to the user the screen position or object they are interacting with, but on the other it allows the light pen’s position to be tracked by the computer. Each observer, user and computer, benefits differently from the presence of the cursor, and it thus needs to be

1.1. Two Important Input Devices for Pointing

tuned to assist both at the same time.

Light pen technology is not in use anymore. CRT-based line-drawing systems are hard to find because they have been displaced by raster devices that refresh the whole screen at the same time instead of one shape at a time. Rendering now takes place in a virtual canvas that is copied all at once many times per second onto the display, which means that we cannot use the same interrupt-driven mechanism to get position information from the shapes being drawn on the screen. We know that direct pointing is still a desirable interaction method, as evidenced by the prevalence of touch sensitive surfaces in all kinds of displays these days, but now it is performed differently. Instead of a light sensor we now use capacitive surfaces, infrared beams or other hardware to do it. Unfortunately, in a room with a large display that is potentially out of the user's reach, direct touch is not a viable option.

Stepping back a bit, we see that the light pen was basically a light sensor, like a camera, mounted inside a pen-shaped holder and connected to the computer. A style of interaction similar to it is conceivable for remote pointing in a classroom. When users need to interact they point a camera at the screen and specifically ask to acquire a cursor. The interaction then takes place, after which the cursor goes away because it is not needed anymore. The ideas embodied in the light pen remain valid and inspirational. If the camera is in direct communication with the processor in charge of rendering the contents of the visual display, we could recover position information for pointing provided there is enough “stuff” to see in the field of view of the camera. The question is *what* needs to be displayed for the camera to get this position information. This will be discussed in detail in the next chapter where we present the chiroptic tracker inspired by early light pens.

1.1.2 Mouse

The mouse is perhaps the best known pointing device because of widespread adoption for desktop computers. It was developed by Doug Engelbart and William English at the Stanford Research Institute (SRI) in California in

1.1. Two Important Input Devices for Pointing

1964, and presented to the world in December of 1968 in what became a very famous demonstration of new ideas for human-computer interaction. It was a successor to the Trackball, a pointing device that consisted of a ball nested inside a mechanism that detected its rotation to move the cursor on the screen. The original mouse consisted of a hand-sized box with two perpendicular wheels attached to its bottom face that translated their rotation into X and Y displacements via rotary potentiometers. A replica from the Computer History Museum is shown in Figure 1.3.



Figure 1.3: A replica of the original mouse by Engelbart and English seen from its side and bottom. Images borrowed from the Computer History Museum’s webpage.

The idea to turn the trackball upside down so that the ball rested on a planar surface came in 1968 from a group in Germany led by Rainer Mallebrein, and independently again in 1973 from a group in Xerox PARC led by Ronald Rider. Ball mice were basically the same technology as the SRI original, but improved the experience of the user: the ball transferred its movement over a surface to two wheels inside the casing that translated it into screen coordinates. Another improvement came in 1981 from two independent teams, one led by Steven Kirsch and another by Richard Lyon, through the use of an optics system that read displacements in a pattern of shapes as the mouse was moved over it. The advantage the optical mouse had over the mechanical one was that the latter usually picked up dust and dirt from the surface where it was used, eventually degrading the moving parts inside and damaging the mouse. A disadvantage of the optical mouse

1.1. Two Important Input Devices for Pointing

was that it needed the presence of the pattern to work. Almost 20 years later, in 1999, Agilent Technologies developed a new version of the optical mouse that got rid of that limitation by constantly taking “pictures” of the surface where it was being used and comparing them in sequence to determine the magnitude of movement. Optical mice are very popular devices these days, with an LED mounted on the bottom arranged next to a lens that refracts its light to shine on the working surface, so that the camera can pick it up. A laser diode can also be used, which improves the contrast of elements in the surface to make the mouse work almost anywhere.

We can simulate the workings of an optical mouse by setting up a camera to look at a desk at a constant angle while we move it over its surface, comparing successive images of the desktop to compute a translation value, and converting that into a command for cursor displacement. Our simulation would run much slower than the mouse, but the underlying principles that make it work would be the same. Optical mice are highly optimized hardware that works with very low resolution cameras, sometimes only 18×18 pixels, with a sampling rate of hundreds or thousands of frames per second. Figure 1.4 shows the elements of the optics system for a mouse. Mice achieve such speeds by using low resolution images and doing computations in the mouse itself, so that when we move the mouse we perceive an instantaneous change in cursor position on the screen.

Engelbart’s 1968 demo presented many important ideas that have been very influential in modern interfaces, from video conferencing to hypertext, and the mouse became the tool that enabled what has been called “direct manipulation”, a technique by which the user can point and interact graphically with objects shown on the screen, and affect them directly by pressing buttons, to do things such as selecting, dragging or highlighting. It was a crucial part in enabling a new metaphor for human-computer interaction. To a lesser degree but in the same vein, we believe that remote pointing is an enabling technology for powerful interaction metaphors with large displays that can change how interactions take place in classrooms.

The mouse was designed for personal use on relatively small displays, and requires the user to be fairly stationary so the mouse can rest against a

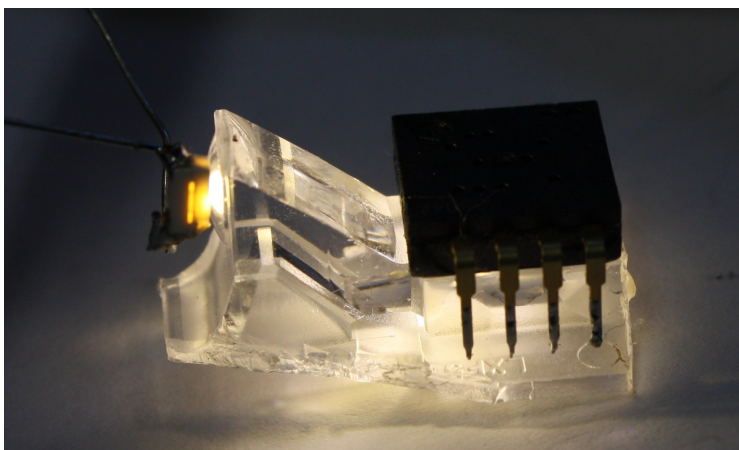


Figure 1.4: Optical system for a mouse. A light source, in this case an LED, shines through a plastic lens to illuminate the surface, while a small chip that contains a tiny camera processes the images to detect translation changes from one frame to the next. Image by Jeroen Domburg from spritesmods.com.

surface while the user operates it. In that sense it fits very well on a desktop with a personal screen, but not necessarily in a room with a large shared display where the user is roaming around. A mouse is usually used as a relative pointing device: each time we move it on the desk the cursor moves relative to its previous position on the screen, not to an absolute position determined by the mouse location on the desktop. A mouse is also the prime example of clutching: lifting it from the desk we can prevent the cursor from moving, and setting it back down in a more advantageous position brings it back into motion.

The cursor used in personal computers is often an arrow with the tip indicating the point of interaction. It is designed to be small enough that it minimizes occlusion of targets in the screen while still remaining visible to the user who is controlling the cursor. A user that loses sight of the cursor will often move the mouse around to find it again, which works due to our increased visual sensitivity to moving objects (see Ware [50]). This is an easy trick that most mouse users have learned (perhaps of necessity) on their own. It helps balance the ease of visually finding the cursor when it

1.1. Two Important Input Devices for Pointing

is lost with the designer's desire that the cursor blend out of the way when not needed. Once users have found the cursor they can track it without difficulty because they are the ones moving it, and so they know where to look for it; but a passive observer will probably have more trouble following the cursor and thus the interaction that is taking place. A shared setting should rethink the cursor design to improve the experience of the audience if that is an important goal of the shared interaction.

The most common type of mouse is restricted to two dimensions by design: it works only when on a surface. This has a few advantages. For example, optical mice only need to be concerned with movement in two dimensions, so the computations from frame to frame are easier to perform and the lens can be optimized for the fixed distance of the camera to the surface. Another advantage is that by placing buttons on top of the mouse it can be used as an input device for selection: the buttons require the user to press on an axis perpendicular to the surface, so clicking them does not cause the cursor to move. A six-degree-of-freedom freehand camera will have a greater challenge interpreting movement change, and placing buttons on it can potentially cause the user to accidentally change the pointing position while clicking a button.

A new user of the mouse has to learn to control it. Some users are so accustomed to the mouse that they can operate it without much thought, but initially they were not as proficient and only became skilled with time. For many it is a worthwhile investment, so it is important to recognize that proficiency with an input device may not be automatic. A simple exercise shows that we are not particularly good at using a mouse without visual feedback. Try the following: take a look at a computer screen with a mouse attached, notice the position of the cursor and choose a target some distance away, make a mental image of what you are seeing and close your eyes, then move the mouse to the target. Research by Phillips and Triggs [37] suggests that you will probably miss. This also happens outside the computer. Try the same exercise with objects on your desk: reaching for one of them with your eyes closed will probably make you miss, knock it over, or at best slow you down considerably.

But humans are adaptable and they learn new skills over time. Our sensorimotor system automatically adjusts to our movements based on feedback from the senses to help us reach our target, which is a fundamental part of why the mouse works so well: we do not even have to make a conscious effort to do it. However, for this to work there should not be too much delay between the movement and the feedback or our error correction mechanism starts to degrade, as research by MacKenzie and Ware [27] suggests.

Mice are ubiquitous pointing devices that for many years have provided tremendous value for our interactions with computers. They will likely continue to do so for some time. They were designed for a different purpose than the one we are interested in, interaction with a large display in a classroom setting, but the insights gained from this brief review will be very valuable to us.

1.2 Contributions

This thesis presents the following four primary contributions:

- *A description for an architecture of chiroptic trackers.* The processes involved and the challenges that have to be overcome to design them are presented. This high-level view provides insights and advice for anyone interested in studying the technology. It forms a framework that allows many possible implementations.
- *A proof of concept implementation of a chiroptic tracker is presented, demonstrating that it has a realizable, low-cost instantiation that requires only a consumer-grade webcam.* We provide details and clarification on many points that might not be obvious from the architecture's abstract description, and we analyze the weaknesses still present in the prototype along with ideas on how to improve it.
- *Results from a controlled experimental study that compares the proof-of-concept chiroptic tracker to a mouse provide baseline information on performance.* The study reveals that the tracker is not only feasible, it also works well for remote pointing tasks in the classroom setting.

The results verify Fitts's Law models of movement time for pointing tasks, which can be used by HCI researchers and interface designers.

- *A design brief explaining the key ideas desirable when building a classroom interface using direct pointing with available hardware.* The design can be realized with the chiroptic tracker or any other device that provides similar functionality.

1.3 Overview of the Thesis

In this first chapter we have presented a comprehensive introduction to the topic of the thesis and background information on the lightpen and the mouse. The lightpen is the inspiration for the chiroptic tracker; the mouse is both the current dominant technology and the target for replacement. Chapter 2 provides further background. It discusses related work directly influencing ours, or that presents alternative approaches to solving the problem we are tackling.

The design and implementation of the chiroptic tracker is presented in detail in Chapter 3, where we also discuss its limitations and opportunities for further development. In Chapter 4 we report the results of the controlled study performed in a classroom at the University of British Columbia, where participants performed a Fitts's Law-style pointing task with our camera-based chiroptic tracker and also with a mouse for comparison.

The research presented in this thesis validates the chiroptic tracker in various dimensions. In Chapter 5 we discuss how a novel classroom interface could be designed using it, considering the lessons learned in previous chapters. Lastly, Chapter 6 summarizes the work that has been presented and lays down a path for future work.

Chapter 2

Related Work

The idea of using the camera in a cellphone to control a cursor is not new. Madhavapeddy et al. [29] proposed enhancing widgets on the display with markers similar to the ones used for tracking in augmented reality environments, so that a user wielding a camera cellphone can twist, move or otherwise interact with the widgets by pointing at them. Rohs [40] went into detail about how this kind of marker can be used to detect position using the camera, describing the algorithms involved. Their markers are very similar to the two-dimensional barcodes that are a popular tool to encode a text string, and so in addition to giving an estimate of the camera's six-degree-of-freedom position, they can be used to tell apart different targets or displays.

Ballagas et al. [5] extended this work to create an interaction called “point and shoot”, where users aim through their phone screens at a target of interest on a large display, and by pressing a button a grid of markers is shown for a few moments while the camera captures a picture, which is then processed to decode the markers and decide what object they are pointing to. After this is done, the grid goes away and the user can continue interacting with the selected object using other means. While the same marker grid could be used to provide cursor tracking, the markers occlude most of the screen due to their size, so they are not ideal. Similarly, an approach like the one by Celozzi et al. [10] to detect camera position, using markers like those proposed by Fiala [12], can be an accurate tracker but is not optimal due to its denseness. Ballagas et al. also described another technique they call “sweep”, which is an analogous mechanism to the optical mouse. It works by analyzing motion flow of consecutive images, and the user does not have to point at the screen, any surface will do as long as it has recognizable features. Both of their techniques were reported to exhibit a 200ms delay

on the phone at the time, and selection with “point and shoot” can be done in around five seconds, but we should keep in mind hardware is faster now, so these times have probably improved.

Decoding position information from shapes is an idea similar to the “structured light” technique used by computer vision researchers, where a known pattern is projected onto a surface and the result analyzed by a camera to determine properties such as shape and depth. Our problem is simpler because we are interested in a planar surface, but the ideas are insightful and will contribute to our discussion in Section 3.4.

Salvi et al. [41] present a survey of pattern encoding techniques. Temporal encoding consists of sequences of patterns that change through time. A camera is carefully synchronized with the projector to pick up the way the pattern is distorted as light falls on the scene, and from that extracts a three-dimensional approximation of what it sees. Spatial encoding uses a two-dimensional pattern such that the neighborhood of each projected point is unique and can be recognized by finding corresponding points. Direct mapping projects shapes or numbers directly onto the scene that encode position. This is the most similar strategy to the marker-based approach discussed earlier.

Displaying the patterns for structured light can be very intrusive, but there are ways to make it invisible to the human eye, as explained by Fofi et al. [14]. One idea is to use infrared projection, which can be picked up by a camera but not by humans. A different approach uses high-speed projectors to quickly switch between the pattern and its negative, so that a camera with a very short shutter time can see it, but our perceptual system will fuse both patterns into a solid grey color. The main disadvantage we see with using structured light as described is that it requires equipping the classroom with more specialized equipment than what is commonly available right now, however, it might become an interesting line of research as this type of hardware becomes more common.

There have been efforts to do feature-based tracking using a camera without the need of extra visual clutter from markers. Jeon et al. [20] propose a range of techniques to do cursor manipulation using a camera phone, one

of which is called “marker-cursor”, where as the name suggests the cursor is the marker. They use a square marker that allows them to calculate a coordinate transformation function (a homography), and the interior of the marker is a triangle that both provides a point of interaction and a way to fix the orientation of the homography. Jiang et al. [21] use a different approach without the use of any specialized markers, cleverly taking the last two frames of the camera and using the cursor displacement to compute a different transformation function, an affine transformation, which they use to approximate the cursor position. Their approach is less powerful in that it only provides two-dimensional translation and rotation information, and when the cursor is relatively stationary the rotation information cannot be computed.

Both of these approaches suffer from the limitation that there is no recovery mechanism when the user moves so fast that the camera loses track of the cursor, meaning they have to go back to where it was left and begin dragging it again. This is an inherent limitation of the “marker-cursor” approach. More recently, Baldauf et al. [3] have adapted more advanced vision techniques to do feature-based tracking of a scene in real time with a camera phone, which does not need any markers at all. They describe a general framework for multi-user and multi-screen interaction that works by computing a homography from a baseline image provided by the screen, and matching of features on the camera image. This is a promising approach that gives full six-degree-of-freedom information of the camera pose, and is reported to work at interactive rates (although no numbers are provided). The challenge remains of how to do cursor tracking on a blank screen, for example for drawing applications.

Computing the camera pose from consecutive images of a scene is one aspect of estimating *optical flow*, the apparent change in position of image elements through time; brightness patches can change from one frame to the next for reasons other than motion, so observing optical flow does not necessarily mean a movement happened. By imposing certain constraints on our interpretation of a scene, algorithms can build a geometric model from optical flow data to estimate the position of the camera relative to

the scene. This model can be used to find the point of intersection of the camera's optical axis and some other object, such as a large wall display.

One way of computing optical flow is through feature-tracking algorithms, which as their name suggest match individual points, edges or corners across different images. An example is the Lucas-Kanade algorithm [24], which makes three assumptions: matching points look the same in every image (they have the same brightness), movements are relatively small, and points move similarly to their neighbors. These assumptions provide enough constraints for an equation that estimates the magnitude and direction of movement from one image to the other, and by assuming small movements the time required to compute the solution is kept manageable. Some image features can be tracked better than others, specifically textured regions work better than plain regions, so the presence of trackable features is important when estimating motion. The technique presented in Chapter 3 is simpler because it assumes the presence of a specific pattern of features, that they all lie on a plane and that they do not move, so any movement comes from the camera: the algorithm was designed to take advantage of these constraints.

There are a variety of alternative techniques that can be used for interacting at a distance with large wall displays. The survey by Ballagas et al. [4] is a great resource for camera-based techniques, including direct and indirect pointing. An example is C-Blink by Miyaoko et al. [30] that consists of a screen mounted camera that is trained to look for color sequence patterns on cellphone screens. Users can move their cursors on the screen by running a program that flashes one of these patterns, showing it to the camera as they move their arms around. An easier approach is to use the touchscreen of a smartphone as a sort of remote control for the cursor, as illustrated by Deller and Ebert [11], where dragging on the phone's screen moves the mouse on the large screen. This removes the benefits of direct pointing, but works on any display. Gallo et al. [16] implemented an algorithm that uses the phone's camera to track the hand of the user in front of it, and the tip of their fingers becomes the cursor position. They did this at an impressive 30fps on a camera phone with a very limited processor by today's standards, but the algorithm's performance suffers with increased

background clutter.

Another common approach involves more advanced camera systems, like those used for motion capture by Vicon [47] and OptiTrack [19], and consists of equipping a room with various precisely calibrated infrared cameras that track reflective markers worn by users in the room. By placing the markers correctly, one can obtain a model of the user's joints and bones, or of a hand-held wand, which can then be used to compute a virtual point of intersection with the display. The main benefit of this technique is that it is very precise, but it requires a lot of equipment that has to be previously installed and calibrated, and so is a more expensive solution.

Others have used depth-sensing cameras to approximate the model of the user's body and carry out the same task with less precision. Muradov [31] implemented such a system where a Microsoft Kinect camera tracked the user's movements, built a user model, and then computed a cursor on the screen as previously described.

Sharp et al. [42] created highly optimized algorithms that use the same sensor to recognize a hand with high accuracy. By using the joint positions of the index finger we can determine a cursor position on the screen where users point, without them wearing any special equipment. The main disadvantage of depth sensors is that they have a restricted field of vision, only a couple of meters wide, so to cover a large area one needs to use multiple sensors carefully synchronized. This increases the cost of implementation and the system complexity considerably. In addition, the infrared light used by all of these devices may conflict with other technology in the room, like some models of personal response (clicker) systems.

Alternatively, the depth-sensing camera can be held in hand and pointed at the scene. KinectFusion, by Newcombe et al. [33], uses this approach to generate a volumetric surface reconstruction of room-sized scenes in real time, which continuously tracks the six-degree-of-freedom position of the depth camera while it moves around the room. Identifying the large display in the scene, the camera pose can be used to estimate where a user is pointing. As depth sensors become more widely available this technique should be of great relevance for direct pointing.

Laser pointers are commonly used by lecturers in presentations to highlight material on the screen. By using a room-mounted camera that is calibrated for the screen's position and dimensions, a system can determine the position of the laser beam on the screen to enable interactions, as described by Kirstein and Mueller [22].

Olsen et al. [36] provide advice to implement such a system, and also proposed an interaction scheme based on synchronized collaboration with the windowing system to allow the user to press buttons, select items from drop down menus, etc. They suggest that the cursor should change depending on the *mode* of interaction that is going on, and proceeded to test their system to measure its effectiveness. Their system had considerable lag and low sampling rate, due to technological limitations at the time, but worked sufficiently well for most of the purposes they designed. Although not a standard test, they observed that users took about twice as long to perform a series of tasks with the laser compared to the mouse. Among their findings they realized that there is a real problem of hand jitter that makes interaction with small objects difficult.

The jitter problem can be expected to increase the farther the user is from the display. A nice feature of using laser pointers like this is that the light on the screen is an effective cursor, easily seen by the audience and by the camera, and its position is updated without delay. On the other hand this means we cannot use techniques like adjusted gain to improve pointing precision. The laser tracking technique has seen many variations, and is a popular choice because it is relatively low-cost, requiring only the presence of a calibrated camera on the room. It can even be extended for multiple users by using modulated light patterns that help distinguish different lasers as shown by Vogt et al. [49], and some researchers have experimented with infrared lasers which are invisible to the human eye, allowing once more to separate the virtual cursor from the laser beam, for example Cavens et al. [9]. Interestingly, users performed poorly with the IR laser when compared to a visible light laser, which the authors speculate is because of the delay caused by the rendering of the virtual cursor.

Comparisons of devices, like the ones performed in the cited research,

can be carried out in different ways. One of the most frequent is to use the theory developed by Fitts [13] where he describes a model of movement time prediction for different tasks that involve users selecting targets and interacting with them in different ways. It consists of a linear model that takes into account the relationship between width of the targets and the amplitude of the movement, which he summarizes in a quantity known as the *index of difficulty* of the task. His results have been refined by Welford [51] to consider the individual differences in subjects and the separate contributions of width and amplitude to the model, recognizing two distinct phases, one of fast ballistic movement directly affected by amplitude, and one of homing into the target affected more by target width.

Fitts's predictive model has been used by HCI researchers in the past due to its consistent reliability, and is now called Fitts's Law. MacKenzie [25] gives a good introduction of how Fitts's tests are carried out, and Soukoreff and MacKenzie [45] present guidelines for standard practices. Furthermore, Shoemaker et al. [44] compare different variants of Fitts's Law and conclude that for pointing at a distance on large displays, particularly when gain values are manipulated, two-part models based on Welford's analysis work better.

Myers et al. [32] performed comparisons of camera-tracked laser pointers and other devices, including the mouse, for pointing tasks from across the room in a manner similar to what MacKenzie and Jusoh [26] did for other remote pointing devices, and found that the laser pointer performed about one and a half times slower compared to the mouse. They also reinforced what was known about the problem of hand jitter affecting laser pointer precision. To the best of our knowledge, feature-based trackers that use markers have not been compared in this way to a baseline device like the mouse.

Chapter 3

Chiroptic Tracker: Camera-Based Remote Pointing

The goal of this chapter is to describe a technique that has potential to be used effectively for remote pointing in a classroom setting without introducing any extra resources into the classroom using only technology that is already there, and to explain the main challenges that have to be considered by designers who build similar tools. The basic idea is to have a pattern of shapes shown on the screen that encode position information. A camera takes pictures of this pattern and then analyzes them to extract information and thus understand where it is looking. In this way a user holding the camera can “point it” at specific targets of interest on the display to initiate interaction.

We begin by specifying a set of design constraints that help guide the creation of the camera tracker, then describe the process of chiroptic tracking in the abstract, what systems are involved, and what are their roles and dependencies. This high-level view of things helps to understand the general concepts behind our tracking technique. We proceed to then look at a specific implementation made to serve as a proof of concept, and compare it to other technologies and measure its capabilities and deficiencies. We end by going over the current known limitations and problems with our implementation, and propose ways in which they can be avoided or overcome, paving the road for future research in this area.

3.1 Design Constraints

From our analysis of the problem, and looking at design lessons from other input devices, we came up with a set of design constraints that helped guide the construction and implementation of the chiroptic tracker. We will summarize them briefly and then explain in more detail what the rationale

is behind each one.

For a classroom setting. The design should consider that the tracker is being designed for use in a classroom setting. That is one of our main objectives, so if it does not work there it is a bad design. Specifically a solution should acknowledge the fact that there are multiple spectators involved, the lecturer and the audience, and that they have different roles. There is a natural imbalance of control in the roles and the tracker is meant for use mainly by the lecturer, however, the audience should not feel lost or abandoned while the lecturer is using the device. A solution should also consider how people will use it. We expect an instructor to have infrequent interactions with the material that is being displayed on the display, and more frequent interaction with the audience, so the device should enable these interactions without getting in the way. Because pointing interactions are infrequent, it is okay if they take a little more effort than with other devices, as long as they keep the flow of the lecture going.

Use existing hardware. There are plenty of technological resources already available that are not being fully exploited. Our belief is that a solution should be possible using only the existing resources, so ideally no more hardware should be required. Specifically, we want to avoid having to introduce special lighting (such as infra-red) into classrooms, or cameras that must be mounted in the classroom, or modified high-frame-rate projectors beyond the standard ones that are commonly found in today's classrooms.

Focus on pointing. An interface meant for interacting with material on a large display involves tackling many different challenges. The tracker itself should only worry about addressing the problem of pointing at targets of interest. If it can also be used for other things, like selecting (clicking), dragging or gesturing, that is fine, but that functionality should be secondary to the main objective, which is pointing.

3.2. The Camera Tracking Process

Flexible. The device should work by itself, without depending on other installations or third party software. Specifically it should be independent of whatever program is used to display slides or other content on the display. It should be possible to adapt it for other uses, so it is desirable to make it as general purpose as possible. It could seem a slight contradiction to require both that it work in the classroom setting and that it is as general purpose as possible, but that is not the case. While it should be designed to work in a classroom first, a solution should not be constrained to work *only* in a classroom. Any solution should allow common techniques used in other pointing input devices, such as clutching and gain control.

Intuitive. Users should be comfortable using it, and it should not get in the way of how they want to naturally interact. It should work as they expect, although some minimal training on how to use it is OK and full proficiency may require a bit of practice. It should not require any calibration. It should “just work”.

The rest of this chapter explains the general idea for the chiroptic tracker and discusses an implementation. We believe our design meets most of the constraints set forth for it. In Chapter 4 we report the results of a controlled user study where we compared it with a mouse to get an idea of how well it performs, and in Chapter 5 we present a design brief on an interface for classroom interactions that uses it as a pointing input device.

3.2 The Camera Tracking Process

There are three systems involved in making the chiroptic tracker work: the display system, the chiroptic sensor and the user’s sensorimotor system, as diagrammed in Figure 3.1. They work together in two loops that make up the camera tracking process. The first loop is between the system display and the chiroptic sensor, and its purpose is to update the position of the cursor based on where the sensor is being pointed at. The second loop is between the user’s sensorimotor system and the chiroptic sensor, and

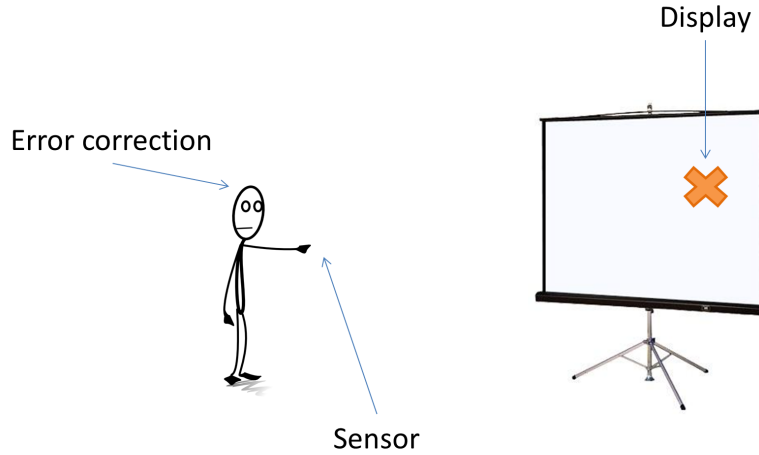


Figure 3.1: The three processes involved in chiroptic tracking. The display renders the cursor and screen contents, the sensor interprets what it sees in the display to extract a position, and the human brain adjusts for errors.

its purpose is to adjust the position of the sensor until the cursor is on a target of interest. We rely on the user's sensorimotor system to make the necessary adjustments to the position of the sensor in order to correct pointing errors without conscious effort, similarly to how we do with the mouse. This is necessary because readings from the sensor are noisy due to diverse factors that will be discussed in Section 3.4, and because there is a delay in the whole system that has to be compensated for. The error correction mechanism is left almost entirely to the human. In our prototype we only provide slight help by smoothing the cursor movement, so we will not discuss the second loop in any more detail except when we later describe the smoothing algorithm. Instead we will focus on the first loop, between the sensor and the display, and discuss how that works. The overall process is as follows:

1. The display system renders a series of shapes on top of the regular content of the screen, including the tracking cross (the cursor). These shapes are called *fiducial markers*² and are used as a reference frame

² Fiducial markers are images that can be tracked with relative ease by vision algo-

3.2. The Camera Tracking Process

that encodes position information for the sensor.

2. The chiroptic sensor is made from a camera and a processing unit. The camera captures a frame that includes enough of the markers that the position information can be decoded. The frame is processed and the sensor relays the position it read to the display system.
3. The display system transforms the position read by the sensor into X and Y coordinates in the virtual display, and then uses that information to update a model for the cursor position, possibly also a model for the shape and position of the markers, and then it starts again from step 1.

The first step of the loop is relatively straightforward. The display should just make sure to show all relevant markers in a way that the camera can see them. The particular design of the markers is more interesting because of how it encodes position information. We will look at a possible set of markers when we talk about implementation in Section 3.3.

It is important to note that the position computed in step 2 by the sensor is *relative to the markers*. The markers create a reference frame that is used by the display system to transform the relative position to absolute coordinates. To get the relative position we usually choose a fixed point in the camera, like the camera center, and then find the relative coordinates of that point with respect to the markers. This is necessary because the camera is capturing a frame out of context, it does not know what part of the display it represents, and does not care about the dimensions of the display or its resolution.

To process the image captured by the camera, the first step is to identify the fiducial markers and make sense of them. These problems are sometimes called *feature detection and extraction* and *scene analysis* and there are many techniques that help address them, from simply looking at individual values of pixels to sophisticated mathematical analysis. See for example the books

rithms. They serve as points of reference and are commonly used to create augmented reality systems or facilitate scene understanding.

3.2. The Camera Tracking Process

by Szeliski [48] and Forsyth and Ponce [15] for an introduction to computer vision techniques. Depending on the design of the markers, it might be faster to use heuristics like geometric relationships to make sense of them.

When the sensor computes a position it has to take into account that the image from the camera will be distorted. It should compensate for this distortion or its reading will be off by a factor depending on how distorted the image is, or just plain wrong if the distortion is too severe. Distortion can come from different sources, including the lens of the camera, but most importantly because of the perspective of the camera relative to the screen, which comes from the angle, distance and rotation between these two elements. To illustrate the point, Figure 3.2 shows an example of two images of the same fiducial markers captured from different camera positions. Note that on the left image in the figure the distance between points A and B is perceptibly bigger than for points C and D, however in the grid those two segments are the same size. That *foreshortening* is being caused by perspective distortion in the image.

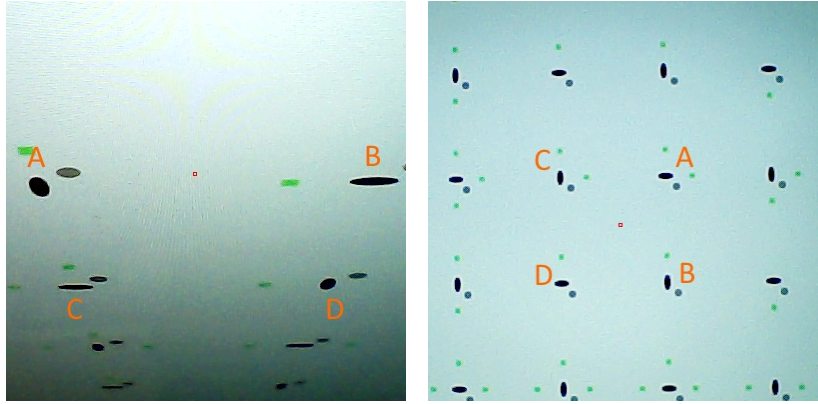


Figure 3.2: Two different perspectives of the same grid cell, as captured by the camera. Corresponding points are labeled with the same letter. The tracker has to deal with the distortions caused by perception. Note also the marked contrast in illuminations on the top and bottom sides of the left image.

To compensate for distance, the relative position can be expressed in “marker units”, which is a unit of measurement derived from the fiducial

3.2. The Camera Tracking Process

markers. It can be, for example, the distance between two of the markers. As long as the display knows the value of that distance in pixels it can transform sensor coordinates to screen coordinates. Compensating for angle and rotation is a little bit trickier because it requires that the sensor interpret the image to understand how the different fiducial markers are arranged, and from that get a transformation function that allows it to undo the effects of the distortion. This is called a perspective transformation, or homography, and computing it is a common problem in the area of computer vision, so there are many techniques that can be used to do it. The previously referenced books by Szeliski and by Forsyth and Ponce are good resources for this.

Step 3 begins by translating the raw reading from the sensor, which is the relative position of the cursor, to an absolute position in screen coordinates. This is generally a relatively straightforward step that depends on how the encoding of the position was done on the markers.

The absolute position can be used to update several models. The first of these is a model of the cursor position that can be used to predict a useful place to draw the cursor, but can also be used for more advanced techniques like target prediction, where one tries to move one step ahead and predict what object of interest the user is ultimately moving to. The second model is for the position of the markers on the screen. When the markers are fixed there is nothing to do, but if the markers move around based on the sensor readings the model can be used to predict good places to render them so that the sensor will see them on the next reading (this is similar to what Sutherland did for the light pen's cursor). The third model that can be useful is for the position in space of the camera, relative to the screen. This position is determined by six parameters, the three coordinates and three angles of the camera, and computing them is known as the *pose estimation* problem in computer vision, which has a strong connection to the problem of *homography estimation*.

There are potential advantages gained from modeling the camera pose. For example, the pose itself is a good proxy of the position of the user in the room, which can be used by interface designers to create interactions that

3.2. The Camera Tracking Process

vary depending on where the user physically located. Another advantage is that by correctly modeling camera pose we can predict where it is likely to move, and so the sensor can filter out readings that are inconsistent with the model, so if the pose of the camera changes from one side of the room to the other in a single frame, then one of those readings is likely wrong.

There is an extra challenge involved when the shape and position of the markers change based on the cursor position, which is that they create what is called a *closed-loop control system* in control theory. We will not go into much detail about this³ except to explain some of the effects that such a system has on the overall tracking process. In a few words, what happens is that the sensor cannot trust what it is reading from the screen because of the delay in the system, and so has to compensate for errors. To illustrate this consider a camera running at 30fps, in a system with a delay of 100ms, with the cursor being sensed stably in the center of the camera. If the camera moves one cm to the right of its current position we would expect a similar movement to be seen on the markers in the display and then the system to return to a stable state where nothing moves. Instead, the first frame after the camera is moved will send a relative movement of a units to the display which will take 100 ms to be seen by the camera again, and by that time the camera will have seen the same markers two more times causing a movement of $3a$ units total. When this error is finally discovered the camera will compensate, but that reading will also get repeated a few frames because of the delay, and so the markers will again overshoot their movement in the opposite direction. This causes an oscillation, which makes the system unstable and is a problem that only gets more complicated if the delay of the system is variable. When the position of the markers is kept in place we do not have this problem because the sensor can be confident that it will always be computing positions relative to the *current* position of the markers. There are techniques such as damping, Kalman Filters or PID controllers that are used to fix these issues, but we will not discuss them here.

³ But see the books by Astrom et al. [2] and Ogata [35] for introductory material to this field.

3.3 Implementation

We chose to make an absolute pointing system with a fixed grid of fiducial markers that are shown persistently on the screen. The architecture for our implementation is diagrammed in Figure 3.3. There are two main processes that communicate in a loop: the display communicates with the sensor by showing a grid of markers that the camera captures, and the sensor communicates its readings to the display using a network socket. Within the sensor process there are several subsystems that carry out the steps required to compute a relative position. First, the camera image is analyzed to extract the markers using a blob detection algorithm, then the blobs are processed to identify four that are the corners of a cell in the grid. The sensor computes a homography from these four points, which is used to identify the rest of the components of the cell and to obtain the position of the camera center relative to it. The relative position and cell row and column numbers are sent to the display process that further transforms it to an absolute position, feeds it to the cursor model, and then renders the grid of markers and the updated cursor according to the model.

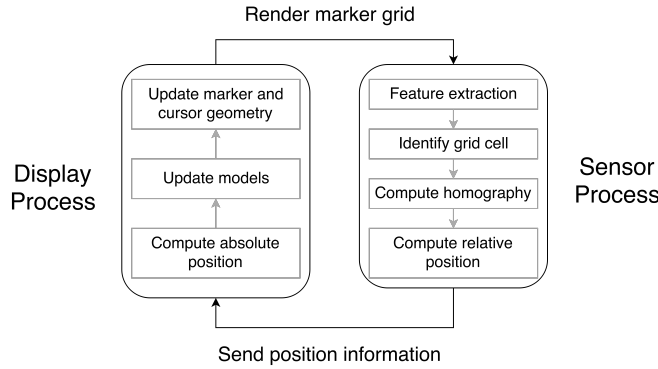


Figure 3.3: The architecture of our tracker’s implementation.

This implementation makes no assumptions about the camera itself, which can have any sampling rate or lens distortion on it. We are ignoring the radial distortion of the lens because we will be focusing on the center of the image, where the effect is minimized, and because for the cameras

that we tested the negative effect on accuracy is very small and likely easily compensated by the user. The sampling rate of the sensor is independent of the refresh rate of the display, so the model for the cursor can be used to update its position even when the sensor readings are relatively infrequent.

3.3.1 Grid of Markers

The fiducial markers are arranged in a grid as shown in Figure 3.4. Groups of four ellipses with alternating horizontal and vertical orientations make up one cell of the grid. There are circular grey markers that indicate the correct orientation of the cell and green markers that encode its row and column number. We use black ellipses for the corners because these are relatively easy to find, and they are the first objects that the sensor will look for to start making sense of the image. By following the main axis of an ellipse we can identify others in the same line or column of the grid. That information is very helpful in deciding which ellipses belong to which cell.

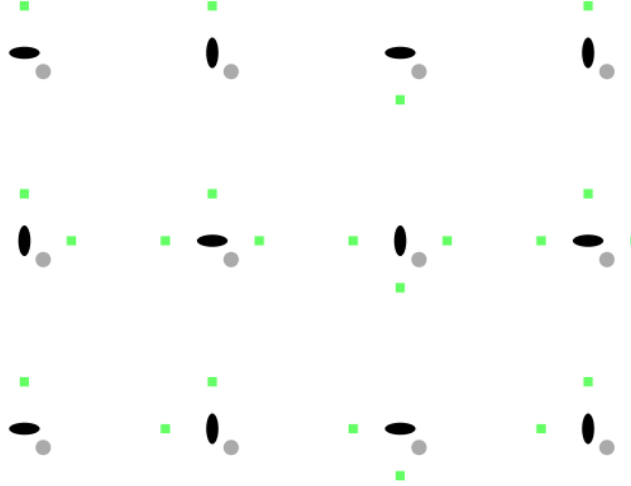


Figure 3.4: Grid of fiducial markers. The black ellipses define a reference frame, the gray circles determine proper orientation, and the green squares encode row and column number.

With this design the camera only needs to be able to see four ellipses that

3.3. Implementation

make up one cell of the grid to ensure that it can compute a position relative to that cell, and because the green markers identify the cell uniquely, the display process can translate that into an absolute position. The resolution of the grid should be adjusted so that the camera can capture at least one cell in each frame while avoiding making the markers so small that they will not be seen by the camera correctly. The size of the individual markers can also be adjusted to improve sensing.

Each cell has unique row and column numbers, represented as bits by the green markers. If a green marker is present, it counts as a 1, if it is not, it counts as a 0. The four horizontal green markers are used for column number and the vertical ones for the row, however we exclude row and column 0 because of an artifact of motion blur: when the camera image gets blurred the ellipses can still be read sometimes, but not the green markers, and the sensor incorrectly reports a movement relative to cell (0,0) which causes the cursor to suddenly jump on the display. So we end up with a design that supports a grid with a resolution of up to 15×15 cells. A final detail comes from the observation that contiguous cells share sides, so we need to arrange the numbers in a way that ensures the two high-order bits of one are the same as the two low-order bits of the next. One possible bit arrangement is shown in Figure 3.5.

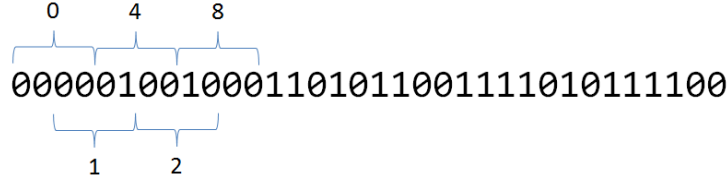


Figure 3.5: A sequence of bits used to encode row and column numbers. Numbers from 0 to 15 are arranged so that the two high-order bits of one are the same as the two low-order bits of the next. Each of the 16 numbers appears once in the sequence.



Figure 3.6: Design of the cursor for the chiroptic tracker.

3.3.2 Cursor

Our cursor design is shown in Figure 3.6. Its shape and size vary depending on its recent position history, with the idea that both the lecturer and the audience can follow it better. Previous research by Po et al. [38] shows that for pointer interaction orientation-neutral cursors or cursors aligned with the direction of movement generally work better, so our design for the resting cursor is a circle and for a moving cursor we show a trail, which also facilitates observers following it with their gaze. The trail gets longer the faster the cursor moves and disappears when it moves slowly. The size of the circle also changes dynamically, expanding with fast movements and shrinking when the cursor is relatively stable. In this way we expect observers can track it more effectively when it is moving, but it will be small enough to afford precise selection when users dwell on targets. A final element of the cursor is an orthogonal cross made up of a vertical and a horizontal segment, which appears only when the cursor is fairly stationary and is meant to enable pixel-precision readings of its position. The specific sizes, shapes and colors in the cursor were determined empirically. Our only recommendation for now is that they should be clearly visible to the humans in the room, but invisible (or easy to tell apart and ignore) to the chiroptic sensor.

For our implementation the cursor is always rendered close to the actual position that the user is pointing. We smooth the actual position computed from the sensor using an exponential moving average, which is a weighted interpolation of the current sensor reading and the previous average, defined by the following equation:

3.3. Implementation

$$C_t = \alpha \cdot P_t + (1 - \alpha) \cdot C_{t-1} \quad (3.1)$$

The value of α is a parameter that can be adjusted to change smoothness. For values closer to 1, the actual current position is weighted more heavily than the history, and so there is less smoothing. Values closer to 0 will make the cursor behave very smoothly, but movement will feel sluggish.

3.3.3 Feature Extraction

The image captured by the camera is just an array of color values with entries for every one of its pixels. To make sense of it we have to first extract features, like shapes or corners, that we can use to conduct a higher level analysis. For our simple grid we chose to implement a naive blob detection algorithm that looks for the ellipses and filters everything else out. It works by first creating a copy of the original image where every pixel has been substituted by just black or white based on a threshold luminance value, so if the pixel has a very dark color it will be changed to black, and if it is below the threshold, it will be colored white. This *binarized image* is then scanned one line at a time looking for segments of black pixels, which are grouped together to form *blobs*. Those are our best guess at identifying the ellipses, so blobs that have a very small or very large area are filtered out as noise. Note that for this to work we require a strong contrast between the black color of the ellipses and the rest of the contents of the screen. This is one of the weaknesses of our method, and we will come back to discuss it at the end of the chapter.

As we find the blobs we also compute their bounding box, their perimeter and the direction of their main axis. The bounding box is the smallest rectangle parallel to the X and Y axes that contains the blob, and if the blob is an ellipse then its center is a good approximation to the center of the ellipse. Blob pixels that are adjacent to some other white pixel are marked as part of the perimeter, then they are sorted by their distance to the center of the blob and with the ten that are the farthest we find a regression line of best fit, which will give us an approximation to the direction of the main

3.3. Implementation

axis of the ellipse. Figure 3.7 shows an example of a frame captured by the camera, its binarized version, and the perimeters and main axes of the blobs that get identified with this method.

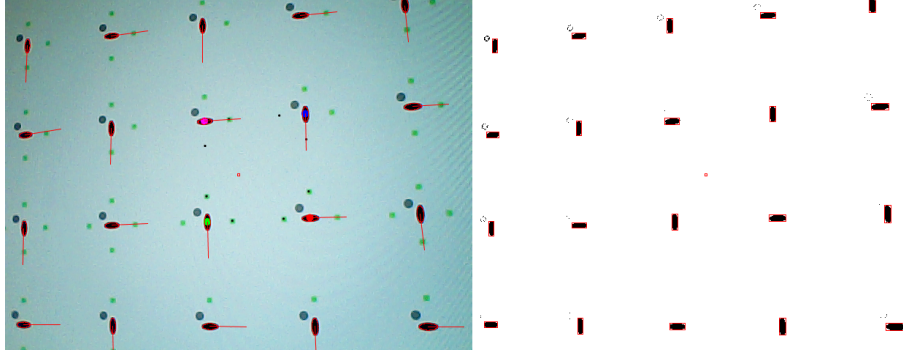


Figure 3.7: The results of feature extraction on one frame of the camera. On the left is the original image with perimeter pixels and the main axis of each ellipse highlighted in red. On the right is the the binarized version used to find the ellipses, and their bounding boxes.

There are more advanced computer vision algorithms that perform robust feature detection with subpixel precision, which would improve the quality of the sensor readings. An advantage of our approach is that it can be done very fast and we can tune it to our specific needs, for example exploiting the properties of the ellipses. Our technique is weak because it requires lighting conditions to be optimal in the room for the tracking to work, so it should be considered only as a proof of concept with much room for improvement.

3.3.4 Computing Relative Coordinates

The next goal is to identify four ellipses from the previous step that form the four corners of a grid cell, which is challenging due to the perspective distortion in the image. Note that in Figure 3.7 the main axes of the ellipses form a staircase pattern, and one “step” of this staircase forms a triangle that is half of a cell of the grid. Using a few heuristics we can identify such a triangle and get the fourth point from that. First, we assume that the

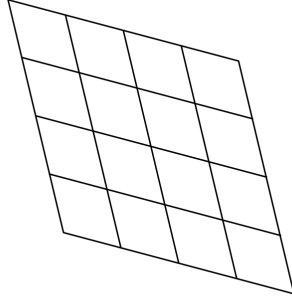
3.3. Implementation

blob that is closest to the camera center is part of the triangle. Then we compare the distance of all other blobs to the line formed by the original ellipse's main axis, and choose the one that is closest. Similarly, the third blob is the one closest to the axis line of the second blob. All comparisons are made using the blob centers, and if there are any ties we break them by choosing the blob closest to the previous blob. In this way we find the triangle we were looking for, and now we only need to identify one final blob to complete the grid cell.

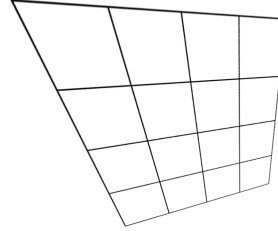
With the three ellipses found so far we can compute a function known as an affine transformation that can undo other forms of distortion. The difference between an affine and a projective transformation is that the former preserves parallel lines and the latter maps them to lines that intersect at a point, so the affine transformation can be used to fix a distortion of our grid like the one shown on Figure 3.8a, but not like the one shown in Figure 3.8b. In a way, however, the affine transformation is a cheap way to approximate the perspective distortion of the image, as shown in Figure 3.8c. We use it to compute an expected position for the fourth point of the cell and choose the blob that is closest to that. Building the affine transformation relative to the cell corners is straightforward if we arrange them as in the figure, and define the position of point A to be $(0,0)$, of B to be $(0,1)$, and of C to be $(1,0)$, giving us two vectors that form a basis for the cell's coordinate system. By mapping the respective vectors in image coordinates to those in cell coordinates the affine transformation follows, and its inverse can be used to get the expected position of point $(1,1)$ in image coordinates.

Now that we have the four points that make up the corners of a grid cell, we can use them to estimate the perspective transformation, which can be used to reverse the distortion of the camera, by using an algorithm known as Direct Linear Transform described in the book by Hartley and Zisserman [18]. This algorithm requires that we provide the correspondence of four points in image space to their coordinates in real world space, so by using a similar idea as before we can map point A to $(0,0)$, point B to $(0,1)$, point C to $(1,0)$ and point D to $(1,1)$. We use the OpenCV library [7] to solve the required system of equations and get in return a homography

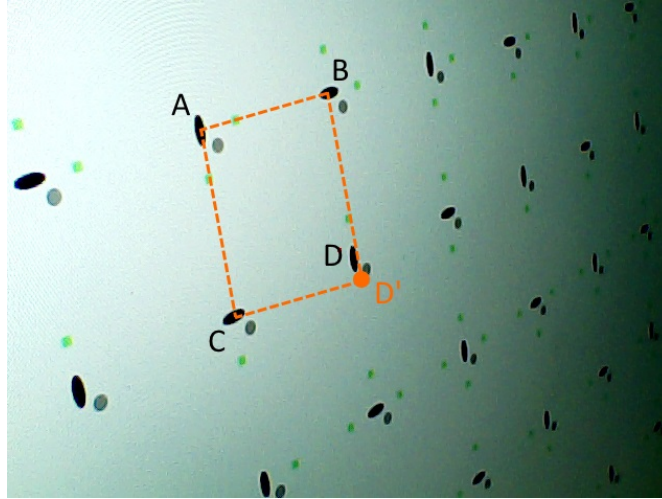
3.3. Implementation



(a) Affine distortion



(b) Perspective distortion



(c) Affine approximation

Figure 3.8: (Top) Difference between affine (a) and perspective (b) transformations. (Bottom) Although the affine transformation based on points A , B and C is not good enough to predict the position of point D , it finds a good approximation in point D' .

expressed as a matrix, which when multiplied by a point in image coordinates gives us the corresponding point in grid coordinates relative to the cell. We can also perform the opposite transformation using the matrix inverse. With the inverse we look for the grey circle in the position where we expect to

3.3. Implementation

find it for each of the four ellipses, and once we find it we know the correct orientation of the grid cell.

Using the inverse of the homography again we look at the positions where we expect the green markers to be. If we find a pixel value that has more green saturation than red or blue, we consider it to be a 1. If not, it is a 0. Putting together the eight green markers in the correct order we decode the row and column numbers of the cell. We then use the homography one last time to transform the position of the camera center to grid coordinates, and send this relative position with the grid row and column numbers to the display process. With that, the sensor process is done and will go through everything again when a new frame comes from the camera. The display process uses its knowledge of the layout of the grid to perform a final transformation to the sensor's reading, from cell-relative coordinates to absolute screen pixels, which it uses from there on.

3.3.5 Cursor Position

Every time the display process gets a reading from the sensor, which happens asynchronously, it uses the absolute coordinates to update its model of the cursor. Our current model is simple, it keeps its current position separate from the sensor reading, which is treated as a goal to reach eventually. In a separate processing thread, when the display process renders a new frame, it asks the model for a position where it should show the cursor. The model takes its previous position and the newest sensor reading and interpolates a value between the two using equation 3.1 to smooth the movement, storing the result for the next interpolation. As long as no new readings come from the sensor, the model will keep doing these interpolations so that the actual cursor position approaches its goal a little at a time, but when a new reading comes the goal is updated and the next interpolation will cause a movement towards that. This simple model works surprisingly well using low values for the α parameter in the interpolation. For a display refresh rate of 60fps we determined an empirical value of $\alpha = 0.2$.

The cursor model has one other function. From time to time the sensor

3.3. Implementation

will get confused, either due to noise, motion blur or other problems, and will report an erroneous position. The model is configured with a threshold value so that if the distance between a sensor reading and the previous one is too big, it will be counted as a fluke and ignored. This heuristic gives stability to the cursor on the screen, however if the user makes a drastic movement that is legitimate, the model will incorrectly filter out the new sensor readings. For that reason these “erroneous” readings are stored in the model and if after a few of them the cursor seems to be stable in a new position, the model updates to move there. We currently filter only one reading, so if the sensor reports the same general position information two times in a row, we update. The values used for this filtering depend on how reliably the sensor can identify the markers in any given setting, and will probably not be needed when more robust vision techniques are used.

In our implementation, when the sensor cannot read position information for any reason or gets a reading wrong, the cursor simply continues inching towards its goal position and then stops. This means that when the user changes position quickly the cursor will seem to “stick” for a moment and then shoot towards the new position, causing a very perceptible delay in response. We tried an alternative model using second order prediction (velocity and acceleration) to keep moving the cursor past its goal, but without imposing some synthetic deceleration, similar to the effect of friction, it created awkward cursor movements. Using the friction it behaved very similar to the simple model, so we did not pursue that idea any further.

The position obtained from the model is used to render the cursor directly. We keep a history of cursor positions that we use to vary the size and trail of the cursor, as explained before. Other implementations could look to use position information in different ways, for example choosing not to show the cursor but instead highlighting a predicted target of interest. The important point to keep in mind is that the computed position does not have to be the rendered position, they may serve different purposes if it helps improve the usability of the interface.

3.3.6 Performance

We implemented this prototype using the Processing 2.0 language, which is based in Java and uses the OpenGL library. The algorithms used are not particularly optimized, so they can probably be tuned to perform faster. Running both processes on an Intel i7-4710HQ CPU @2.5GHz with 12GB of RAM and an NVIDIA Geforce GTX 850M graphics card, the sensor code can run at slightly more than 240fps processing a 640×480 px resolution image each time. In the lab we measured the delay from the time the grid is shown to the cursor position being updated at around 150 ms.

This is high-end equipment, but we expect that the same techniques should run on more modest resources at adequate sampling rates, so in principle it should be feasible to use something like a smartphone directly for processing. The biggest problem might be delay in getting the image from the camera to the software, and the restricted memory available. Lowering the resolution of the camera would also improve time performance, and it is possible that users could deal with the lower sensor accuracy. All of this is speculation and remains to be tested, but we feel confident these ideas can be implemented in current hardware.

3.4 Known Limitations

Our implementation is enough for our purposes — it is a proof of concept showing that the idea is realizable, it works well enough under controlled conditions that we can perform a study to compare the chiroptic tracker to a baseline device like the mouse, and measure its performance to get an idea of its capabilities. It is still an early prototype implementation, and as such has some problems and limitations that have to be addressed in future research. Here we discuss the more pressing ones.

3.4.1 Occlusion Caused by the Grid

The markers that are displayed for the sensor have to be shown on top of the contents of the display, which causes the obvious problem of occluding

important information, and they are distracting for the audience. For the classroom setting this problem is mitigated by the assumption that the lecturer will have only sporadic interactions with the display, and so the grid only needs to be shown during those times. In Chapter 5 we present a design brief for a classroom interface that uses the tracker as it is now, so we believe that it can be made to work and that the benefits outweigh the cost.

Nevertheless there are things that can be done to fix or mitigate the issue. The simplest is to adjust dynamically the resolution of the grid and the size of the markers depending on the position of the camera, so that it can see them as needed but at the same time keeping the density of markers to a minimum on the screen. Making the grid change dynamically introduces again the problem of dealing with a closed-loop control system that was discussed earlier, but it is a common problem with proven solutions in many engineering applications. Alternatively, we could create better prediction models for where the markers need to be, leaving only the four that are necessary to compute a homography and in this way removing most of the grid from the screen. In the extreme case we could make the cursor itself be made up of markers. This is also a closed-loop solution that we are starting to look into.

If the projector in the room has the capabilities, the grid could be shown in infrared light so that the audience cannot see it, and a camera without an infrared filter would still make an effective sensor. Or if the camera has high enough framerate and the projector can work at 120fps or faster, we could show the grid on only *some* frames and not others, or switch its colors in alternating frames so that they “cancel out”, in a way similar to the invisible structured light techniques discussed in Chapter 2. In theory, when done fast enough the audience would not even notice the presence of the grid. We feel these are weaker solutions because they depend on the availability of less common hardware, and in particular a blinking pattern could end up being more disruptive than a grid that is always on, while infrared could already be in use by a different system in the room, such as some models of clickers, however, they are still possibilities to look into.

3.4.2 High Color Contrast

The algorithms we use are not robust, and the sensor simply stops working when the colors on the display have little contrast. That is often the case when using projectors that have a weak light source, or with poor color balance settings. Another issue is ambient room light coming from sunlight or artificial light sources, which has the effect of attenuating the contrast of colors in the camera image. This is particularly bad when a light falls directly on the surface of the display, because the same rendered color will have different RGB values in the sensor’s image depending on how much light falls on it. The human perception system has *color constancy*, which allows us to identify two very different patches of light as the same color (for an example consider the sun falling directly on half of a desk, even though the half in shadow and the half in the light look very different when taken individually, we know the desk surface is a single color). There are many techniques in computer vision that model color constancy, see for example work by Agarwal et al. [1] and by Gijzen et al. [17], so this is a problem that can likely be solved. As a partial solution the sensor could synchronize with the display to show a pattern that helps calibrate the expected color values, which would happen only at the beginning of the interaction or from time to time during normal operation.

Another alternative is to be less reliant on color, for example by using distinct shapes instead of distinct hues, or by using a different technique for tracking altogether. For example, we could perform point correspondence across consecutive frames directly on features found on the contents of the display, without having to show any extra markers, and get a homography from that. In a situation where there are not enough features on the display, like a drawing application with a blank canvas, we could show a gentle, neutral texture as a background that presents enough recognizable features to the camera. Szeliski’s [48] discussion on motion estimation algorithms would be a good starting point to go deeper into this.

3.4.3 Chaotic Movement

Natural hand jitter, noise in the camera sensor and motion blur can all cause erroneous position readings that send the cursor flying around the screen chaotically, which is very disruptive when trying to interact with the display. Some of the techniques we mentioned in the implementation, like smoothing and filtering out extreme changes in position, help to alleviate the problem at the cost of making the cursor feel somewhat more sluggish. Instead of smoothing we could try manipulating control to display ratio values, which would lower the impact of jitter. That would not help with problems caused by motion blur, for which smarter and more robust algorithms could be designed. Ultimately, better cameras with a higher sampling rate and improved sensors would get rid of many of these issues, so perhaps as hardware continues to evolve this limitation will fix itself.

3.4.4 Lens Blur

When the image from the camera is not properly focused the blur can cause the sensor to misbehave. If the user is consistently interacting at a sufficient distance from the display this is not a problem, because most lens systems can focus objects at far distances accurately without major dynamic adjustments. Otherwise the tracker should have some way of dealing with this, like the auto-focusing features found on some cameras.

3.4.5 Lag

The 150 ms it takes for the sensor to capture an image and decode the cursor position, together with the smoothing techniques used to improve accuracy, make the system feel slow to respond. In Chapter 4 we will present the results of a study designed to measure the usability of the device as it is, but it would always be desirable to have less delay. This could be accomplished by using different software tools with faster access to the camera, and eventually we could manufacture dedicated hardware like it is done with the mouse today. In the meantime, an interesting approach is to create other

models that help predict the user movements, either by keeping track of the six degrees-of-freedom of the camera pose in the room or by other predictive techniques.

3.4.6 Acute Angles

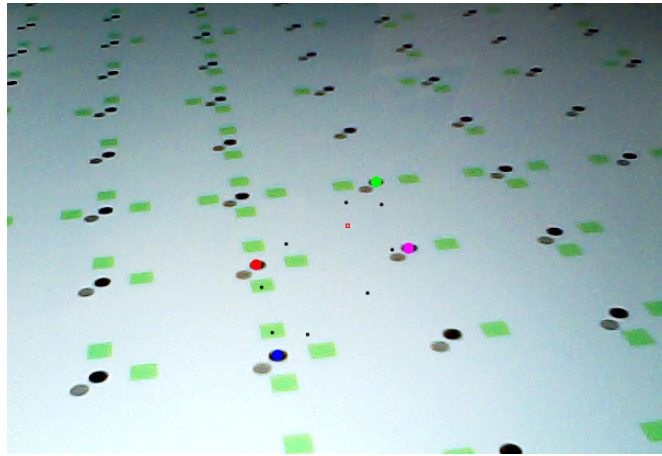


Figure 3.9: An early version of the chiroptic sensor that incorrectly identified four blobs, highlighted with red, green, blue and magenta, as the corners of a grid cell. The cause is extreme perspective distortion in the image and the use of weaker heuristics for grid identification.

When the distortion from perspective is too strong, our heuristics fail and the sensor reports erroneous information. Figure 3.9 shows an example from an early version of our prototype that used different heuristics for detecting the grid cell, where the sensor got confused and labeled four blobs incorrectly as the corners of a grid cell. When the angle is too extreme the distortion is simply too much and the sensor will just not work, but there is a limit point where it sometimes works and sometimes not, and that can be frustrating to the user because the cursor moves erratically on the screen. The use of better heuristics or a different marker design could help address this problem.

3.5 Pilot Testing

We piloted the study described in the next chapter with seven colleagues and found many interesting observations that were used to improve the experiment. We consider them valuable lessons when designing other similar studies, and also because they give insight into how users might interact with chiroptic trackers. We summarize them here.

Mouse gain was set to a value that all participants considered comfortable enough to reach all targets yet giving them enough precision for smaller movements. Windows 8.1 has a ten notch slider for adjusting mouse gain; after trying different values all seven pilot participants suggested independently that the 6th notch was optimal for the task.

Our first design of a camera tracker used a glove in which all fingers except for the index had been cut out. The stripped-down plastic casing of the camera was attached to the tip of the finger with hot glue and the cable allowed to hang loose from it. Users reported that the weight of the camera was enough that they were getting tired by the middle of the study. From observations, it was also clear that the flexibility of the glove and weight from the cable made the camera droop from the end of the finger, so that users had to compensate by pointing higher than what was their intuition. We decided to drop the glove design and instead create a device similar to a laser pointer that was used for the study. It is possible that pointing using just a finger is a feasible idea, but clearly doing it well is a design challenge of its own.

We reduced the number of trials per width and amplitude combination from 16 to 12, incremented the duration of mandatory pauses between block of trials, and increased the frequency of optional pauses. This was because several of the participants in the pilot study mentioned that they were getting tired before the end. We also adjusted the *side* position to be difficult yet doable by all participants, bringing the user a little closer to the screen and with a more obtuse angle than what we had originally planned. The final position was nearly 4 meters from the screen center and roughly at a 40 degree angle, a smaller angle caused significantly more trouble for most

users.

The grid of fiducial markers was adjusted so that the camera could perceive it from all three positions correctly. We used a grid of 6×3 cells, with ellipses of 24.4 cm of major axis and 9.75 cm of minor axis. We adjusted the smoothing factor α in equation 3.1 to 0.2 so that the cursor position was the weighted sum of 20% of the most recent position as reported raw by the sensor and 80% of the previous average. This was evaluated subjectively by pilot participants as a good trade-off between smoothness and responsiveness.

We had observed before that new users of the camera tracker tend to “drag it” a bit cautiously as if attempting not to lose it, while in reality this is not necessary because it is an absolute pointing device and will find itself on the screen even if it gets temporarily lost. This observation was corroborated with pilot participants. To try to encourage participants to take advantage of the affordances of absolute pointing, we experimented with a lower refresh rate for the cursor, which updated its actual position at full sampling rate, yet only provided updated visual feedback every 150ms. This seemed to work, but qualitatively all pilot participants reported a strong preference for the smooth cursor. We resolved the dilemma with another observation: after dealing with the narrow targets that were farther apart, users usually discovered on their own that jumping to arbitrary positions was safe and effective, so we designed practice sessions to start with precisely those targets.

A final problem with the study design came from one of the tracker’s limitations. As mentioned previously, in certain situations like those caused by motion blur, the tracker algorithm gets confused and decides the user is pointing at a radically different position. This causes the cursor to fly around the screen in chaotic movement. While the problem was minimized with thresholding, it still happened from time to time. This would cause some pilot participants to lose the tracker completely and spend a long time, often tens of seconds, trying to reacquire it. We asked them to experiment with different grips, the two more common ones being the *screwdriver grip* and the *pencil grip*, as shown in Figure 3.10, and the pencil grip seemed to be the

3.5. Pilot Testing

one where they could reacquire the tracker faster. Accuracy and precision with both grips seemed comparable, so we decided to ask participants to use the pencil grip as a requirement for the study. We should mention that a *pistol grip*, similar to how a person holds a gun, was suggested as a superior option, but we did not pursue it because it could potentially make people uncomfortable. We also observed an effect, possibly transferred from mouse use, where participants would move the camera around when losing the cursor in an attempt to find it again. This would often result in motion blur in the camera, causing more chaotic movements and confusion for participants. We preempted this problem during the study by explaining a better strategy for finding the cursor, which is described in the next chapter in Section 4.3.4.



Figure 3.10: Two of the grips used to hold the chiroptic tracker. The *screw-driver grip* on the left is commonly used to hold a laser pointer. The *pencil grip* on the right seems to be slightly more intuitive for aiming the tracker.

These observations suggest that usage of the tracker is not always intuitive, and users will benefit from understanding the general principles behind it, as well as from better design that takes into account these natural interaction affordances.

Chapter 4

Comparing Remote Pointing to the Mouse: a Study on the Feasibility of Chiroptic Devices

Having described how to implement the camera tracker, we now turn our attention to evaluating its performance and comparing it to what is probably the most common input device used in classroom presentations today, the mouse.

Our goal is to obtain initial models of user performance for the camera tracker, and in the process attempt to demonstrate that the camera can be a valid device for pointing at a distance. To do this we designed and ran a study in which we compared the camera to the mouse in a standard Fitts's Law task [13].

One of the clear downsides of our current approach to the camera tracker is that the grid of fiducial markers has to be rendered on top of the screen contents, which causes occlusion of potential targets of interest and possibly confusion to the user. For this reason we were also interested in measuring the effect of the grid on the ability of users to carry out tasks with targets of different sizes. Our design assumes that prolonged presence of the grid on the display will become "background noise" that users will learn to "filter out" after a while.

A secondary goal of the study was to collect data that might be used to inform the decisions of designers of interfaces that utilize the camera tracker.

4.1 Hypotheses

Based on our observations while developing and testing the camera tracker, we expected it to perform worse than the mouse, but still well enough that it can be considered an effective alternative input mechanism. The study tested the validity of the following hypotheses that each addressed some

aspect of performance:

- H1. Myers et al. [32] showed laser pointers are around 1.5 times slower and have slightly higher error rates than the mouse for tasks requiring pointing at a distance. Due to the similarities with the laser pointer, we expect that the mouse will perform better than the camera, with both a better throughput and a better error rate, but the camera's performance will only be a factor of 2 or 3 times slower than the mouse and 10-25% less precise.
- H2. There will not be an effect on user performance with the mouse when the grid is on for relatively big targets (those targets that the fiducial markers can only occlude partially).
- H3. Due to the nature of the computer vision algorithms employed, participant hand jitter, and motion blur effects, pointing with the camera tracker will be less effective the farther a user is from the screen and the more acute the angle to the screen is.
- H4. Pointing performance with the camera while sitting down will be comparable to pointing with the camera while standing up.

4.2 Empirical Models of Pointing Performance

To test our hypotheses we used empirical models of pointing performance drawn from the literature. All are based on the well-known Fitts's Law, first published in 1956 [13]. Fitts proposes a linear model for movement time that takes into account the *index of difficulty* of a movement task, which is a quantity that depends on the ratio of the amplitude of the movement and the width of the target (Equation 4.1). His original formulation (Equation 4.3) has been refined by Soukoreff and MacKenzie [45] to better match Shannon's theory of information, which in part inspired Fitts's model, using a different index of difficulty (Equation 4.2). The resulting model (Equation 4.4) has

4.2. Empirical Models of Pointing Performance

become an accepted standard for movement time studies.

$$\text{(Index of Difficulty)} \quad ID = \log_2 \left(\frac{A}{W} \right) \quad (4.1)$$

$$\text{(Shannon Index of Difficulty)} \quad ID = \log_2 \left(\frac{A}{W} + 1 \right) \quad (4.2)$$

$$\text{(Fitts)} \quad MT = a + b \log_2 \left(\frac{A}{W} \right) \quad (4.3)$$

$$\text{(Shannon-Fitts)} \quad MT = a + b \log_2 \left(\frac{A}{W} + 1 \right) \quad (4.4)$$

Both of these movement time models are called one-part formulations because they consider only the ratio of amplitude and width—their individual magnitudes are not relevant. In 1968, Welford [51] recognized that there are two phases of movement, one of rapid ballistic motion affected mainly by amplitude, and one of homing into the target which depends mainly on target width. Welford’s proposed a two-part formulation (Equation 4.5), which explicitly recognizes the different contributions of width and amplitude. Recently, Shoemaker et al. [44] suggested a fourth formulation (Equation 4.6) that combines aspects of the Shannon-Fitts one-part formulation and Welford’s two-part formulation, which they used to analyze pointing tasks at a distance on large wall displays.

$$\text{(Welford)} \quad MT = a + b_1 \log(A) - b_2 \log(W) \quad (4.5)$$

$$\text{(Shannon-Welford)} \quad MT = a + b_1 \log(A + W) - b_2 \log(W) \quad (4.6)$$

The b parameter in Equations 4.3 and 4.4 is the rate of change of movement time as the index of difficulty is varied. Looking at it helps to understand how a particular model behaves. It can also be compared across different models, for example for different pointing devices, to understand their relative performance, but care should be taken because this comparison does not take into account the a parameter of the models. Instead, Soukoreff and Mackenzie [45] suggest a comparison based on *throughput* per individual, which they define as the average of ratios of effective index of

difficulty over movement time. Throughput is measured in bits per second, in keeping with the information-theoretic interpretation of Fitts's Law and its variants. For participant i in a study performing in conditions indexed by j

$$TP_i = \left(\frac{1}{n} \sum_{j=1}^n \frac{ID_{ij}}{MT_{ij}} \right) \quad (4.7)$$

where n is the number of target conditions (combinations of target width and amplitude). The average of all participant throughput values is the overall throughput for a given condition. This value can be used to compare conditions directly.

Welford's also suggested that using the actual width of targets for the computations might be incorrect. For example, a person moving quickly towards a wide target will tend to tap it on a position closer to the near edge than the far edge of the target, and so the distribution of tap positions will likely fall on a region narrower than the target's width. The *effective width* of a target is the region where most observations happen, irrespective of its actual width. A similar reasoning can be used to derive the concept of *effective amplitude*. Soukoreff and Mackenzie [45] encourage the use of effective widths and amplitudes for Equations 4.3 and 4.4, and we adopt this for our analysis of all model formulations. We use their definition of effective width as the region where approximately 96% of the observations occur, and so it is given by

$$W_e = 4.133\sigma \quad (4.8)$$

where σ is the standard deviation of the end-point positions of the observations for the particular target. Effective amplitude is defined as the mean amplitude between successive observations for the target. We will write simply W and A for effective width and effective amplitude throughout the text, rather than the more cumbersome W_e and A_e .

A thorough discussion of the four empirical models of pointing performance that we will use is provided by Shoemaker et al. [44]. We follow

closely their approach for analyzing the results of our study and for comparing between the models.

4.3 Method

We conducted a controlled study that utilized the one-dimensional horizontal serial tapping task commonly used in Fitts's Law studies in which participants have to point at and select alternating targets on the display. We based the design and analysis on similar experiments in previous HCI research, and we carried out some additional analyses suggested by Soukoreff and MacKenzie [45]. We also analyzed our data using Welford-style two-part formulations as suggested by Shoemaker et al. [44] to learn more about the separate effects of width and amplitude on movement time.

We chose a one-dimensional task instead of a two-dimensional task because we wanted to carry it out in a realistic setting, which for us is an auditorium-style classroom with a large display that is usually located high above and out of reach of the instructor. Such a setup creates an acute vertical angle between the user and the display that causes important differences in target perception due to foreshortening and, in early tests, it was seen to cause discomfort for some users. In the study participants were shown rectangular targets that they had to point to and select in alternating motion, as illustrated in Figure 4.1, switching between devices as the experiment progressed. The full details of the experiment are provided in the remaining sections of this chapter.

4.3.1 Participants

There were 24 participants who took part in the study (7 female) between the ages of 22 and 34. They were recruited by advertising in UBC student email lists and by word of mouth. To avoid experimental bias due to handedness or personal handicaps, all our participants were screened by self-report to be right-handed, as well as having normal or corrected to normal vision, no color-blindness, and to be regular computer users averaging 8 hours a week or more of computer usage.

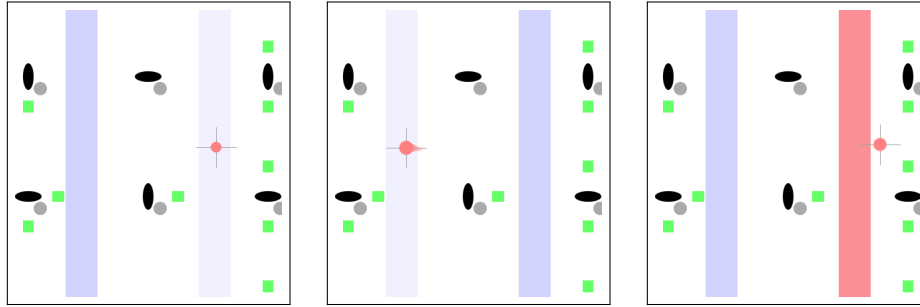


Figure 4.1: Illustration of the task performed by participants: (left) the cursor is moved from its starting position towards the darker rectangle, (center) if the user clicks correctly on the target the other rectangle becomes the new target, and (right) if instead the user misses the target it flashes red and the targets then switch.

The study had approval from the Behavioural Research Ethics Board at UBC. Participants were fully informed of the purpose of the study and of their right to withdraw at any point. They were compensated \$10 for their time.

4.3.2 Apparatus

The study was run in an amphitheater-style classroom, which is the type of room the camera tracker was designed for. The display was 390cm wide and 248cm tall, raised 216cm above the floor. An ASUS laptop with an Intel i7-4710HQ CPU @2.5GHz, 12GB of RAM, and an NVIDIA Geforce GTX 850M discrete graphics card using 64-bit Windows 8.1 ran all of the experimental software. The laptop computer was connected via HDMI to an EPSON PowerLitePro Z8050W projector located at the back of the room. The display generated a 1280×800 px resolution image at 60Hz. The laptop computer was used to record all experimental data.

During each condition participants were located in one of three positions in the room as shown in Figure 4.2. Positions #1 and #3 were closer to the screen and thus at little or no elevation relative to where instructors normally stand during lectures. Position #2 was farther away, where the

4.3. Method

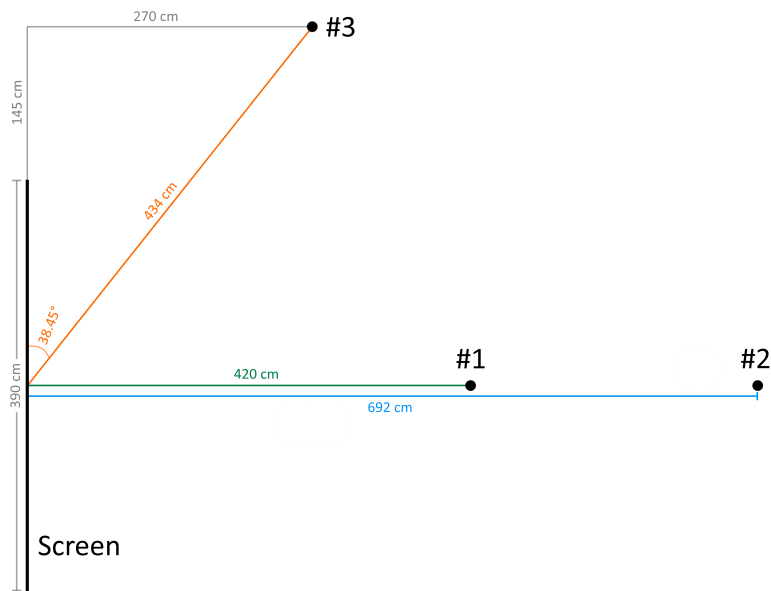


Figure 4.2: (top) A picture of the room illustrating what participants saw. All lights were turned off during the study. (bottom) A diagram drawn to scale of the room layout. Positions #1 and #2 are perpendicular to the center of the screen, and position #3 is off to the side.

floor is significantly raised relative to the front of the room, and so had an elevation of 74cm. Assuming eye-level at 120cm from the floor when sitting down and 170cm when standing up, vertical angles from eyes to the center of the display were approximately 27.5° and 22° for sitting and standing poses respectively in position #1, and 7.9° and 32° for a standing pose in positions #2 and #3 respectively.

One of the main goals of the study was to make a fair comparison between the mouse and the camera, so both were tested from position #1, which had the line of sight perpendicular to the screen and provided an optimal angle of view for both devices. The other positions were only used for the camera. For this reason, position #1 is of main interest and is where most of the study took place.

We tried moving position #1 closer to the screen to increase ecological validity for the camera condition, but this proved too straining for users, particularly in the mouse conditions. Position #1 is close enough to an actual lecturing position that we believe it is a good compromise that allowed users to perform the study comfortably.

Position #2 increased the distance to the screen, so we could get an idea of how this factor affects performance and which formulation of Fitts's Law works better. Position #3 was chosen because it represents more closely how a lecturer might use the tracker in actual practice. It is at the front of the classroom but off to the side, so pointing can be done without fully turning your back on the audience.

We used the OpenCV library [7], written in C/C++, for homography estimation and a Java-based driver for the i>Clicker devices that was developed by Shi [43] and refined by Beshai [6] to capture clicker interaction. The rest of the software for the study was implemented natively in the Processing 2 language, which runs on the Java Virtual Machine and can maintain a refresh rate of 60fps.

Mouse

Participants were given a Microsoft Comfort Optical Mouse 1000, which they could move freely on a desk to reach all parts of the display, and use the left button to select targets. Mouse acceleration was disabled in the operating system, and the gain adjusted to a value at which participants could reach the more distant targets without clutching while still allowing them enough precision for smaller targets. This gain value was determined by early testing with multiple pilot participants. The mouse was connected to the computer by a 6m active USB extension to ensure a strong signal, and the end of the extension was anchored in place so that its weight would not pull on the mouse.

Camera

We adapted a generic consumer-grade low-end USB webcam with a resolution of 640×480 px, sampled at 25fps with automatic exposure and white-balancing that could not be turned off. We removed extra parts from the plastic casing and attached it to a wooden dowel using hot glue to create a device resembling a laser pointer. The camera's USB cable was twisted around the dowel and held in place with tape to ensure its weight would not pull on the front side of the pointer. The camera focus can be adjusted by twisting the screw-mounted lens on the front; it tends to change unpredictably with movement and vibrations, so it was brought to an appropriate state and held in place with a rubber band twisted on itself. Figure 4.3 shows the result, which is what was used for the study. The total cost of this device was less than \$5.

For target selection, we wanted to avoid the problem of the pointer moving out of position when the user clicked a button on it, so instead participants held in the left hand an i>Clicker device that was synchronized with a base station connected to the experiment's computer. Any of the 5 buttons of the clicker could be used to indicate selection. As with the mouse, we used the active USB extension to connect the camera, similarly anchored, except for position #3 where we had no reliable way of doing



Figure 4.3: The physical prototype of the tracker device used by participants. The image also shows the pencil grip they were asked to use during the study.

it. For that position users held the end of the extension in their left hand together with the i>Clicker so the weight of the cable would not interfere with the right hand’s use of the camera.

4.3.3 Study Design

Our study was within-subjects. Experimental conditions for the target rectangles were 3 widths, 24, 48 and 96 pixels — 7.31, 14.63 and 29.25 cm respectively, and 3 amplitudes, 200, 400 and 800 pixels — 60.94, 121.88 and 243.75 cm respectively, as measured from center to center of the rectangles. All 9 combinations of widths and amplitudes were used for the *target* conditions. Additionally there were 6 blocking *pose* conditions summarized in Table 4.1.

The first four poses, all in position #1, made up the main part of the experiment. We used those to determine the parameters for models of performance based on Fitts’s Law, to compare performance between the mouse and the camera, and to measure the effects of the grid being visible when the mouse was used. The order in which participants experienced these were

4.3. Method

Pose	Pos	Device	Details
no grid	#1	Mouse	Sitting down, grid off.
grid	#1	Mouse	Sitting down, with grid visible.
sitting	#1	Camera	Sitting down.
standing	#1	Camera	Standing up.
far	#2	Camera	Standing up.
side	#3	Camera	Standing up.

Table 4.1: The six *pose* conditions used in the study.

fully counterbalanced to account for learning or tiredness effects. Those four conditions were always presented first. After that, participants would do the “far” and “side” poses, which were partially balanced between them so that of the 12 participants that experienced “standing” as the first camera pose, half of them did “far” first and the other half did “side” first. Similarly for the 12 participants that experienced “sitting” first. *Target* conditions were randomized within each block for every participant.

Everyone experienced all 6 *pose* conditions, performing 12 trials for each of the 9 *targets*, so there were $24 \times 6 \times 3 \times 3 \times 12 = 15,552$ total trials. The whole session took about 45 minutes to complete.

4.3.4 Procedure

After reading and signing a consent form explaining their rights and what the experiment was about, as required by the UBC Behavioural Research Ethics Board, participants were presented with a questionnaire to make sure they met the requirements for participation.

They were told they would see different shapes on the display, in particular two vertical blue rectangles, one dark and one light, and that their task was to point at the dark rectangle and click, at which point the rectangles would switch colours and participants would then have to click on the other rectangle that had become dark, going back and forth between them until the end of a block. They were shown both devices, the mouse and the camera tracker, and given a brief explanation on how to operate them.

For the mouse they all had previous experience, and could use it as usual

4.3. Method

pressing the left button to select a target.

For the camera we explained that it was a regular webcam, that they were supposed to hold it like a pencil and point it at the screen, and that for selecting they should hold the i>Clicker in their left hand, using any of the buttons on it to indicate a selection. Other than the pencil grip, they were allowed to hold and move their arm in front of them however they wanted, but were advised that they could rest their arm while sitting down or hold it close and bend it while standing to avoid getting tired. They were told that if ever they lost the tracker, the best strategy was to avoid waving their hand in the air trying to find the tracker again, but instead to hold it still for a moment while pointing at the center of the display and let the tracker find itself.

Room lights were turned off for the remainder of the experiment, so the only light in the room came from the display.

Whenever participants clicked with the mouse or pressed a button on the clicker, the active (dark) rectangle would become inactive (light) and vice versa. If a participant missed a target, the target would flash red for a moment to indicate that an error had occurred. The cursor consisted both of a red circle and an orthogonal cross. Participants were told that as long as the center of the cross was inside the rectangle, it was considered a good tap. They were instructed to emphasize accuracy first and speed second.

Participants carried out practice sessions with the camera and with the mouse before doing the main task until both they and the experimenter felt confident about their proficiency with the devices and their understanding of the task. At a minimum, all participants did 3 practice taps for each of the *rectangle* conditions, all while sitting down in position #1. Two participants requested a second practice session with the camera. For practice with the mouse, visibility of the grid was synchronized with their first mouse condition, so those who would see the mouse with the grid condition first got practice with the grid visible but those who would see the no-grid condition first did not.

After each *block* consisting of all trials for one of the poses, there was a minimum 2 minute pause. During this time the experimenter would switch

devices (if appropriate) and move the participant to a new position in the room as necessary. Additionally, within the block there were 2 optional pauses, one after every 3 *target* conditions. Most participants used the pauses to take a few seconds of rest when using the camera tracker. After finishing all 6 blocks for the poses, participants were presented with a questionnaire to gather qualitative data about their experience with the camera.

4.4 Results

For the analysis the first trial of each *target* condition was thrown out. From time to time during the study, the camera would have a series of bad readings causing the cursor to jump around the screen chaotically. When this happened, participants sometimes had trouble finding it and getting it to stabilize again. For this reason, of the remaining 14,256 trials, 14 were considered outliers for taking more than 10 seconds and were discarded (11 of those were in the “side” condition, all on the left target of the narrowest width and widest amplitude). A further 31 trials were thrown out because they ended 5 standard deviations or more away from the target center.

We encountered a problem during the experiment that was not found in piloting. After pressing a clicker button, the remote imposed a delay of .75 seconds where no other button press would go through. This meant that sometimes when participants clicked one target and then another in less than .75 seconds, the second click would not be registered, causing them to do a “double take” when the targets did not switch, increasing their registered movement time for that trial. We did not remove these trials from the analysis because they were sporadic and difficult to detect reliably, but the fact should be kept in mind when interpreting results.

We measured movement time and tap position for each trial, from which we extracted error rates for each condition. Analyses for these variables and the throughput calculations suggested in the literature for Fitts’s Law experiments are reported in the subsections that follow.

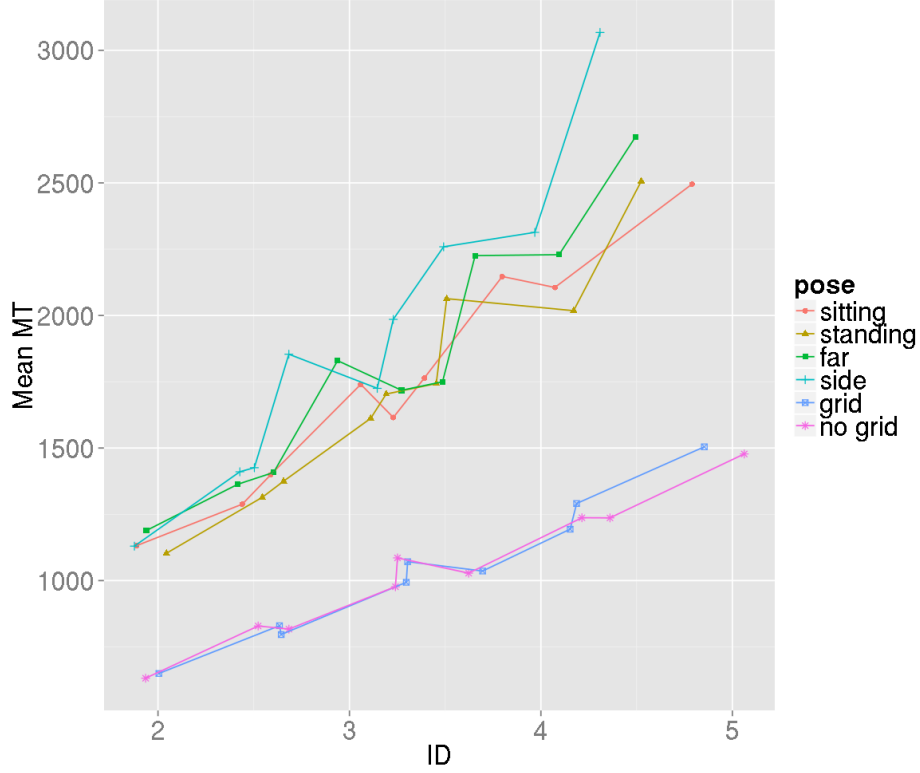


Figure 4.4: Mean movement times in milliseconds at different indexes of difficulty for all *pose* conditions. Lines are included only for readability.

4.4.1 Movement Time

Data were aggregated by *target* and *pose*. Mean movement time, effective width and effective amplitude were computed for pose-target conditions as discussed in Section 4.2. Figure 4.4 shows the mean movement time values plotted against index of difficulty (Equation 4.2) from the Shannon formulation of Fitts’s Law, which is an accepted standard for this type of study.

Shoemaker et al. [44] recommend that two-part models be considered when analyzing Fitts’s tasks with multiple levels of gain. Although we did not vary gain in our study, we did vary distance from the screen, which has also been reported to be better modeled by a two-part formulation by

4.4. Results

Rajendran [39]. This is an area of much interest for current and future research, and so we present results for analyses using all four models discussed by Shoemaker et al., summarized in Table 4.2

	One-Part (Fitts)	Two-Part (Welford)
basic	$a + b \log_2 \left(\frac{A}{W} \right)$	$a + b_1 \log_2(A) - b_2 \log_2(W)$
Shannon	$a + b \log_2 \left(\frac{A}{W} + 1 \right)$	$a + b_1 \log_2(A + W) - b_2 \log_2(W)$

Table 4.2: The four formulations of Fitts’s Law considered for our analysis. Each predicts movement time MT from target width W and amplitude (distance) of movement A .

We also carried out F-test comparisons between pairs of nested models to see if the two-part formulations work better than the corresponding one-part models for our data. The results for Fitts and Welford models are shown in Table 4.3 and for Shannon and Shannon-Welford in Table 4.4.

Pose	Fitts			Welford				F-test	
	a	b	R^2	a	b_1	b_2	R^2	F	p
no grid	287.45	229.82	0.969	657.96	201.23	276.76	0.994	24.60	0.003
grid	207.58	256.68	0.961	496.64	235.68	295.57	0.977	4.10	0.089
sitting	423.91	431.59	0.963	800.18	406.57	484.27	0.971	1.83	0.225
standing	270.28	472.59	0.930	659.64	454.38	540.75	0.940	1.06	0.342
far	343.00	489.98	0.907	1363.62	439.54	661.95	0.954	6.16	0.048
side	130.03	622.24	0.899	770.34	595.10	733.73	0.912	0.89	0.383

Table 4.3: Movement time models for the Fitts and Welford formulations. Significant differences in nested models are highlighted in bold.

Pose	Shannon-Fitts			Shannon-Welford				F-test	
	a	b	R^2	a	b_1	b_2	R^2	F	p
no grid	153.23	257.05	0.970	539.28	225.15	300.44	0.994	24.20	0.003
grid	53.77	288.66	0.966	357.04	265.05	325.39	0.982	5.45	0.058
sitting	150.03	490.01	0.967	540.01	461.85	539.21	0.976	2.17	0.191
standing	-17.90	534.16	0.932	389.13	513.39	601.30	0.943	1.16	0.323
far	20.56	560.55	0.915	1079.70	503.28	727.42	0.963	7.90	0.031
side	-297.77	718.35	0.908	366.29	687.16	827.14	0.922	1.01	0.353

Table 4.4: Movement time models for the Shannon-Fitts and Shannon-Welford formulations. Significant differences in nested models are highlighted in bold.

In every condition the two-part models outperform their corresponding one-part formulations, as measured by R^2 values. By the same measure, the

4.4. Results

Shannon models did better than their respective basic counterparts. The F-tests show a significant difference between one and two-part models only for the “no grid” and “far” conditions. From this data it seems the performance of the different models is generally comparable in our intended classroom setting: both one-part and two-part models work equally well, although the Shannon-Welford model might be slightly preferred.

For one-part models the b parameter is the slope, or rate of change of movement time as the index of difficulty increases. For our range of index of difficulty values, the camera models have a slope roughly twice those for the mouse, which suggests we can expect movement time for pointing tasks to take roughly twice as long to be performed. This however is not entirely clear from the data because the a parameter, the intercept, also plays a role. We will come back to this point when we analyze throughput.

Note that the R^2 values are worst in conditions “far” and “side”, which makes sense because those correspond to tasks with a higher handicap, and are prone to more noise due to the bigger challenge presented to the camera’s vision algorithms. In addition the “side” models are oversimplifying, in that they group taps from both left and right targets together, while in reality tapping on the left side was considerably harder than the right side because of the reduced visual angle due to the geometry.

4.4.2 Error Rates

	24px			48px			96px		
	200px	400px	800px	200px	400px	800px	200px	400px	800px
no grid	4.6%	5.0%	5.4%	2.7%	5.0%	3.0%	1.1%	1.5%	1.5%
grid	3.8%	3.8%	3.8%	1.9%	4.6%	4.6%	0.4%	0.8%	0.8%
sitting	6.4%	8.7%	7.3%	3.4%	3.4%	4.6%	1.9%	2.7%	1.1%
standing	5.0%	9.5%	13.7%	1.9%	5.3%	3.1%	0.0%	1.9%	1.1%
far	11.4%	12.9%	16.0%	3.8%	3.0%	3.0%	1.1%	1.5%	0.8%
side	14.0%	15.2%	19.1%	3.4%	4.6%	6.0%	1.1%	2.7%	3.8%

Table 4.5: Error rates for *pose* and *target* conditions.

Error rates for each *pose* and *target* condition are shown in Table 4.5. Conditions “no grid” and “standing” represent the fairest comparison be-

tween the mouse and camera, respectively, and they have similar error rates except for the more narrow rectangles, where the mouse is more reliable.

There is no clear pattern between the “sitting” and “standing” conditions, however the “grid” condition outperforms the “no grid” condition in all but one case, suggesting there might be a correlation between presence of the grid and improved error rate.

Errors for narrow rectangles in the “far” and “side” condition rise drastically, suggesting that the camera tracker’s reliability falls with increased depth and acute angles, as expected. Hand jitter is probably a factor here, but also the increased presence of chaotic movements due to deficiencies in the vision algorithms.

4.4.3 Throughput

We use the Shannon-Fitts formulation of index of difficulty (Equation 4.2) for the throughput equations described in Section 4.2. The results are presented in Table 4.6.

Pose	grid	no grid	sitting	standing	far	side
Throughput	3.54	3.58	2.04	2.08	1.92	1.79
SD	0.29	0.39	0.29	0.30	0.34	0.32

Table 4.6: Throughput means for all *pose* conditions, in bit/s.

The “standing” and “no grid” throughput values strengthen the argument that the mouse is roughly twice as efficient as the camera as an input technique in this setting. We ran a one-way repeated-measures ANOVA to detect possible effects of pose on throughput. Mauchly’s test revealed no violation of sphericity, $\chi^2(14) = 22.55$, $p = 0.07$, and the results show an effect of pose on participant’s throughput, $F(5, 115) = 571.69$, $p < 0.001$. Bonferroni *post hoc* tests show that there are significant differences between mouse conditions and camera conditions (all $p < 0.001$), between “side” and both “sitting” or “standing” (both $p < 0.001$) conditions, and between “far” and “standing” ($p = 0.01$) conditions. No other differences were significant, in particular no difference between the mouse conditions “grid” and “no grid”,

or between the camera conditions “sitting” and “standing” was found.

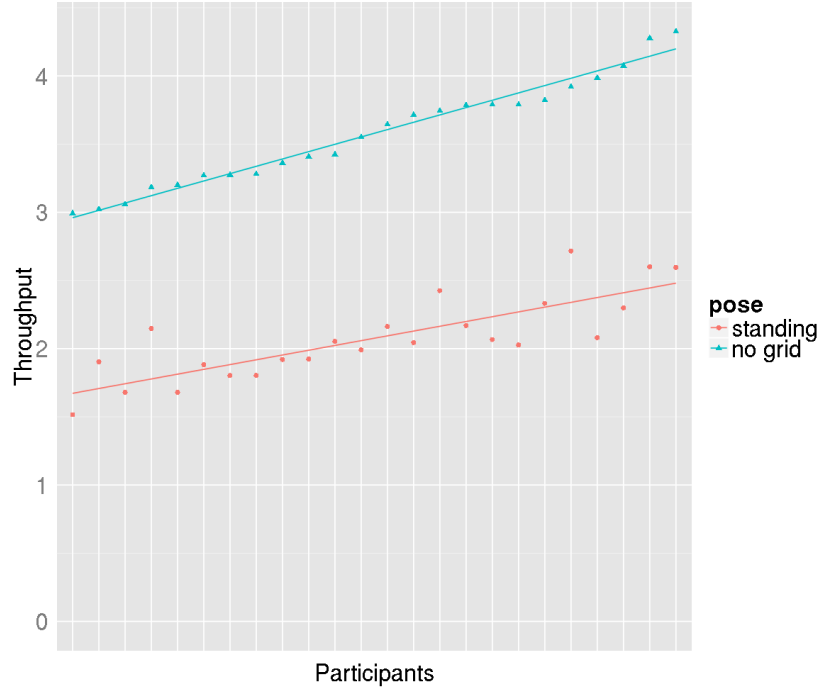


Figure 4.5: Comparison of average throughput values per participant. Participants were sorted by increasing throughput in the “no grid” condition, and the values for the “standing” condition are also shown.

Throughput values varied considerably between participants. Figure 4.5 shows average throughput values per participant for the main mouse and camera conditions, sorted by increasing throughput with the mouse. The data for the camera condition shows a clear upward trend, suggesting a positive correlation between performance with both devices. Only a few participants had markedly different performance levels between devices, performing strongly with the mouse but poorly with the camera, for example. This could be due to individual differences in how users interact with computer interfaces, or to individual strategies while carrying out the task (a couple of the participants with high performance did mention they consciously searched for an efficient strategy).

4.4.4 Subjective Data

We asked participants to rate the level of difficulty they perceived the task to be when using the camera, from 1 (Easy) to 5 (Impossible), to report any particular strategy that they might have employed, and to provide any other comments they had regarding their experience with the task. The average difficulty was rated as 2.292 (SD = 1); the details are summarized in Figure 4.6.

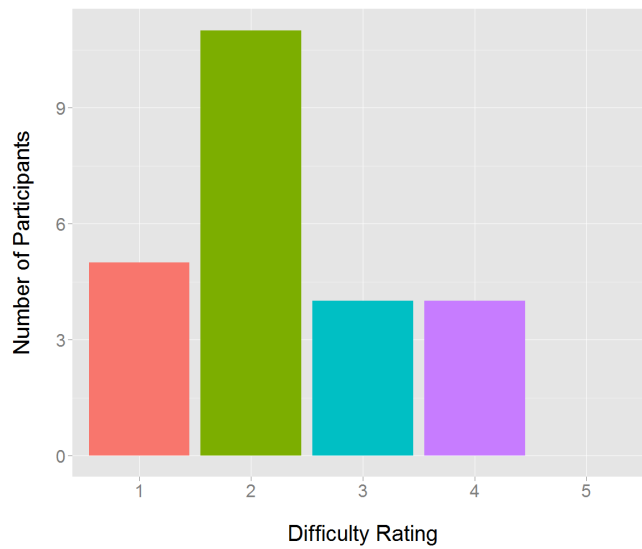


Figure 4.6: Difficulty ratings subjectively reported by participants for the pointing task using the chiroptic tracker, from 1 (easy) to 5 (impossible).

The most common strategy was that of doing a fast initial movement to launch the cursor close to the target, and then fine-tuning their pointing with precise movements, with 13 participants saying they did this. It is possible that the technique was encouraged by the defect in sensing described in Section 3.3.5 that makes the tracker “sticky” when suddenly moving a long distance. Two participants who commented they were trying to be fast used a technique in which they would move quickly and then click while the cursor was passing over the target rectangle, so it did not matter if they overshoot as long as they clicked in time. Three people tried to train their body to

remember the distance they had to flick their wrist to go from one target to the other.

Several participants mentioned during the experiment that lag was perceptible. A few of the participants expressed frustration with the narrower targets, both in their written comments and while doing the experiment. Other comments were about too much shakiness/sensitivity in the cursor, fatigue, preference between devices, and ergonomics. Looking more into these issues would be beneficial, but from this data there does not seem to be a main theme or trend. In the end it might come down to just individual differences and preferences.

- **Cursor sensitivity.** P06 said *“The lack of accurate and stable control makes the task difficult when using the camera”*, P11 said *“The pointer is shaky and too sensitive to movements”*, and P08 said *“Pointing the camera like a pen is probably not the most stable way, holding a flashlight might be more stable figure”*.
- **Fatigue.** P22 said *“Using this pointer for long periods of time may be exhausting”*, and similarly P07 said *“The arm is getting tired after a while, but I imagine in a real-world scenario one wouldn’t do so many interactions after another”*. On the other hand P09 said *“Time was not a factor. I was not tired.”*
- **Preference of device.** Opinions were varied. P18 said *“I liked the mouse so much more”*, but in contrast P12 said *“So far it is a great technique to look and point at objects”*, while P24 said *“Same difficulty as mouse, mostly. Exception is when it flitted all over the place”*, possibly referring to the chaotic movements that happened from time to time when the sensor misbehaved. Most likely, each device has its place depending on the task, as P20 hints, by noting *“The mouse may be more accurate but the camera felt easier and more intuitive to use.”*
- **Ergonomics.** P17 said *“If the camera can be held in [another] way [...] it might be easier and put less pressure on your wrist.”*, and P21 summed it up as *“I think it will be important to take ergonomics into*

account.” We have stated before that the design of a physical tracker is a challenge in itself because we should consider the comfort of users and balance their preference for grip with the benefits of one that promotes better accuracy and precision.

4.5 Discussion

Our results indicate that the chiroptic tracker is a valid technique for a pointing task in a classroom setting. While there is still much work to do, our findings are encouraging. Pointing with the camera seems to follow the Fitts paradigm, and while generally the Shannon-Welford model had the best fit, there was no real difference in between formulations for most conditions.

We used a constant gain value for all of the conditions in the study. Had we varied gain, we might be better able to determine whether a two-part Welford formulation is required, as has been discussed in the literature.

As discussed by Myers et al. [32], users of laser pointers experience significantly more vertical jitter in their hand movement than horizontal jitter. In future research it would be valuable to measure the effects of vertical movements on camera tracker performance.

In closing, we briefly summarize our four hypotheses and the degree to which our experimental data support each of them.

H1 was supported. Error rates between camera and mouse conditions in position #1 are comparable, and throughput values with both devices are within a factor of two of each other. While there seems to be varied preferences in devices, it is clear that participants can perform sufficiently well with a camera tracker. The mouse has the home advantage, because all our participants are regular computer users familiarized with its use. We do not expect users will ever match the mouse performance as they become more experienced with the camera—they are fundamentally different methods of interaction—but at least their subjective experience might be improved.

Good interface design will also increase usability. Narrow targets were

distinctly harder to hit than the rest, so care should be taken by designers to work around the natural limitations that come with pointing at a distance.

H2 was supported. Analysis of throughput shows that the presence of the grid did not impact user's performance with the mouse significantly, and their error rates were comparable. While the requirement to overlay the grid on top of the screen contents is unpleasing, it seems users can work around it effectively in this task. This is a promising finding as we continue to design a classroom interface using the chiroptic device.

H3 was supported. This is not a very surprising result, because a larger distance from the screen increases the effect of natural hand jitter, or, put another way, it reduces the width of the targets when measured in visual angles from the point of view of the user, making it more difficult to acquire them precisely. Sharper angles have the added problem of bringing the vision algorithms in the tracker to their limit, making the sensor less reliable. In the future we would like to perform more experiments manipulating control to display ratio at different distances from the screen to dig deeper into the effects of distance, and improving the sensor is still an active research area.

H4 was supported. There was no significant difference in throughput values for the "sitting" and "standing" condition. The error rates are only slightly larger for the thinnest rectangles at maximum movement amplitude for the "standing" pose.

Chapter 5

Design Brief for a Classroom Interface

In chapter 4 we saw that users perform about twice as fast with the mouse than with the camera tracker, but the camera tracker has a strong advantage in that it allows for direct pointing that gives freedom to the lecturer to move around the room and perform relatively complex interactions without having to go back to the computer. It is this freedom that we believe greatly offsets the performance cost of the camera tracker.

Informed by our experience implementing and using the chiroptic tracker, in this chapter we present some key design ideas for anyone who wants to build a classroom interface with a direct pointing device. These ideas have not been tested yet; our goal is to motivate discussion and provide a starting point for future interface designers.

We will describe our design brief in abstract first, making the case for direct pointing in general, and later we will describe in more concrete terms an example of how our own prototype could be used as part of a larger system to support engagement in an interactive classroom. We present our recommendations as a set of assertions, accompanied by a brief summary behind our reasoning for each.

5.1 Goals of the Interface

Engaging for students. The interface should be a way for lecturers in a classroom setting to interact with a large wall display. Its main goal is to empower users to perform interactions that are more dynamic and complex than what is common today, which is largely restricted to flipping back and forth between slides and pointing at the screen with a laser. Even though lecturers are the primary target, the design should take care to acknowledge the needs of the rest of the occupants of the room. This means, especially,

that the audience should be able to follow along with whatever action is taking place.

Tailored to expert users. University lecturers can be expected to spend some effort in learning the interface, so while it should not be unnecessarily complex or obscure, a rich and powerful interface is more desirable than a simplistic one even if it requires a bit of training and practice to fully master it. Lecturers are professionals. They should have professional-quality tools and they should invest time and effort developing their skills with those tools.

Embedded in current practice. Interactions should flow naturally as part of the conversation between lecturer and audience that happens in a class. Disruptions to this conversation should be avoided as much as possible. When possible the interaction should mimic traditional classroom activity that has stood the test of time. Traditional chalk-board lectures had many advantages. We believe some of these can be reclaimed while still profiting from the many new opportunities digital media provide.

Multimodal and embodied. When people talk, they perform *deictic gestures* that make their words more precise or nuanced. Pointing a finger at something is a strong deictic gesture, as is switching your gaze towards an object: most people will instinctively look to see what is being pointed or looked at. This is the reason why direct pointing is such a desirable quality in a classroom interface, because the lecturer can use body language that the audience understands naturally, and if this body language is translated into updated interface elements, then the conversation will flow effortlessly. The action of turning towards the screen to point is an indication to the audience that they should turn their attention there. In contrast, a pointing technique that manipulates the display indirectly, like a mouse or a touch-sensitive surface that acts as a remote control, does not convey this natural human understanding.

5.2 Required Resources Available Today

We assume a lecturer has both hands available to carry out direct pointing interactions, one to hold the camera and another for pressing buttons. Obviously this will not always be the case (special versions of the equipment will be needed for those with disabilities, which is a challenge we will not address here beyond noting that this consideration is essential to the final design). The setup we used before conforms to this description, with the camera tracker on one hand and the 5-button clicker on the other. In the future, as the physical form of the tracker evolves, it might be possible to have everything integrated into a single device, or to free one hand entirely by placing the camera on the wrist, for example.

We further assume that instructors are more likely to tolerate some level of inconvenience if it grants them a richer set of interactions with students, which is the reason they are already willing to clip on microphones and bring their own laptop computers to class for lectures. Because our direct pointing techniques are intended for instructors giving lectures, not for students listening to lectures, convenience for the casual user is not (yet) a requirement.

A fundamental assumption is that there is a small but sufficient set of distinct buttons available to the user. There has to be at least one button to switch the device on and off (clutching). Two more buttons are enough to do most interactions because we can use one to toggle between modes and the other to trigger actions. Having more than just three buttons reduces the number of distinct modes required, but it also increases the complexity of the device. Our goal is not to provide the user with a mobile keyboard, so we will assume there are about five buttons available, which is what an i>Clicker has and thus is readily available in many classrooms.

Buttons on an i>Clicker can detect only a “press down” event. It makes no difference if you keep a button pressed or let it go, so actions such as dragging, which is usually performed by clicking, moving, and then releasing, have to be done instead by pressing, moving, and then pressing again (i.e., we are restricted to point-and-click rather than drag-and-drop). This simple

5.3. *Styles of Interaction*

“button down” behavior is the absolute minimum functionality of a button, and is the way that many clicker systems work (including i>Clicker).

In our design the camera is part of a dedicated device for tracking. While it is possible that a device such as a cellphone can be used for tracking with the techniques described in earlier chapters, it is not optimal because user accuracy may suffer: the physical form factor of a phone makes pointing unintuitive. In general, it is best to make the interface design independent of the physical properties of the pointing device. As the research progresses, the tracking device could become more sophisticated and versatile, however, this should not be relied on in the high-level design. The simple approach taken in our prototype is sufficient to support a rich set of interactions.

The two pieces of hardware that we require are available today. The i>Clicker fully meets our requirements for the button device. Low cost cameras, configured as in our prototype, provide the tracker device. While these are not currently available as commercial products, the supporting technology is available at commodity prices. This means that in a practical sense all of the equipment is accessible for use in the classroom and ready for commercialization. No additional hardware is required beyond what is already in place in most classrooms: an instructor’s laptop computer (or a built-in classroom computer), WiFi connectivity, and one or more display systems connected to the computer.

5.3 Styles of Interaction

Interactions happen mainly through direct pointing, but the meaning of the pointing gesture changes depending on what buttons are pressed. The interface should not require direct access to the computer’s resources, such as keyboard, mouse or personal screen, unless it is an unavoidable exception such as connecting to the WiFi network or the classroom displays, or enabling the chiroptic tracker support software on the computer.

Interactions are meant to be sporadic. The design of the interface does not require continuous input from the user for prolonged periods of time. The chiroptic device is not the best way to carry out prolonged interactions,

even in the classroom, because that is not what it is designed for. For our prototype, this explains why the grid occluding the contents on the display screen is not a problem: the grid serves as an indication that an interaction is underway, but it goes away immediately when the interaction is finished, so if the interactions are carried out quickly and sporadically, occlusion will not be a significant distraction because the grid will get out of the way and allow the lecture to continue once the interaction is complete.

Usually lecturers will be standing and positioned in front or to the side of the main wall display, with their backs to the display and looking at the audience. A lecturer will want to avoid blocking students' view of the display, so the angle of pointing will usually be considerably skewed. Lecturers will have some room to move around and may instinctively move to a more advantageous position for pointing when it is required.

Students in the audience might have secondary displays in front of them, such as computer screens or paper-based notebooks, but their attention will often be mainly focused on the lecturer and the main wall display.

5.4 General Recommendations

We make four recommendations about appropriate design choices that are particularly suited to the type of classroom-based interaction we want to support.

Modes. These allow a user to perform different actions using the same gesture because the gesture is interpreted in the context of the current mode. An illustrating analogy is the tool palette of drawing software such as Photoshop: clicking and dragging the mouse has a different effect based on what tool is selected. Our prototype uses modes. Some modes could transition automatically from others, and some could share functionality. For example “copying” a section of the display and “highlighting” both require the user to “draw” a region first, so those two actions (modes) could become available after the drawing mode.

Try to minimize mode switching. This will involve determining which modes are more common so they can be activated by default, and which

5.5. Example of Future In-Classroom Interaction

ones are natural transitions from other modes, so the user does not have to switch explicitly.

Make buttons perform actions that are similar across modes. This helps the user build a simple mental model because the button mapping is consistent. For example, the button that switches tracking off could also be used to cancel an action, such as the drawing of a shape, or to reset the position of an object that is being dragged.

Be agnostic about other classroom software. It is acceptable to treat the contents of the display in a special way that is optimized for a classroom interface with direct pointing, but do not tie the interface to a specific software package, like a slide presentation program. Ideally it should work independent of the software responsible for rendering the contents of the display, and the device driver for the tracker should handle its own set of commands and not expect them to be dealt with by plug-ins to other applications. As standard functionality, include a mode that makes the tracker and clicker work like a mouse with emulated left and right-click buttons, so that any software can be used remotely with the tracker. This should not be the default mode of interaction.

5.5 Example of Future In-Classroom Interaction

The recommendations in this chapter were designed in abstract where it was possible, independent of any specific implementation of the chiroptic tracker. In this section we give an example of how an interaction could take place with our current prototype and a 5-button clicker like the one used for the study described in Chapter 4. Our description is only one way in which the interaction can take place; the design space for a classroom interface based on direct pointing is vast. Exploring it more deeply is an interesting topic for future research.

We will consider the following scenario:

James is giving a math lecture discussing the implications of a theorem. His current slide is relevant to the context of the discussion. It includes an equation with multiple parts. A student

5.5. *Example of Future In-Classroom Interaction*

asks a question and James realizes the answer involves only two isolated parts of the equation, so he decides to highlight them. He turns towards the display, presses button 5 on his clicker to activate the camera tracker he is holding on his right hand. This action overlays a grid of fiducial markers on the display. He points at the first relevant section and presses button 1 on his clicker, which starts drawing the outline of a rectangle at the current cursor position. Then he drags the cursor to enlarge the rectangle until it surrounds the first part that he wants to highlight in the equation, and then presses button 1 again to complete the rectangle. He does the same thing to create another rectangle surrounding the second part of the equation, and then presses button 2, which causes the two rectangles and their content to remain bright while the space outside them becomes darker, achieving a highlighting effect. At this point the grid and cursor disappear, the camera is no longer active. James turns back to the student to explain the relevant concepts. When the question is resolved he presses button 4 on his clicker, which removes the highlights and returns to the original state of the slide, and he continues his lecture.

In this example a few things stand out. First, the interaction was not planned, it was improvised based on the dynamic requirements of the classroom. Second, the tracker and clicker only come into play during the actions taken by the lecturer to highlight regions of the screen. The rest of the time they are out of the way and the lecture proceeds as normal. Third, it is possible to carry out this example with our technique, and the resources to do so are available to an instructor now.

Designing an effective classroom interface that is based around direct pointing is a task that requires iteration and experimentation. We hope the discussion in this chapter provides enough insights to begin the next steps in the process. Ultimately, the camera-based pointer and the interaction techniques that use it will be only a part of a larger multi-user interface for

5.5. *Example of Future In-Classroom Interaction*

all the people in the classroom, an interface that might resemble an orchestra of smart instruments playing in coordination to the mutual benefit of the room's occupants.

Chapter 6

Conclusions and Future Work

Technological resources in modern classrooms are not being used to their full potential. They could be leveraged to create a richer interface for the benefit of both lecturers and students. A key component to enable such an interface is a pointing device that allows direct manipulation of content on the screen without tethering users to the computer and ideally without introducing more hardware into the classroom.

We have explained how such a device can be created by using a pattern of structured shapes on the display to encode position information, paired with a hand-held camera that when pointed at the screen interprets the pattern and determines a position to draw the cursor. In addition, we provide an implementation of these ideas running reliably at sampling rates that are high enough to provide a smooth experience for users of the pointing device. Our implementation is still a proof of concept with many possible improvements, including the use of better vision algorithms for feature extraction that are more robust to varying lighting conditions and reduced color contrast. There is a vast pool of knowledge in computer vision that we have not fully explored, so it is possible that a completely different approach would work better, for example by tracking feature points of the display itself.

It would also be interesting to explore the creation of a closed-loop version of the device, where the shapes on the display are moved around depending on the user's location, to both minimize clutter on the screen and ensure position information can always be obtained. As long as we track four distinct points, we should be able to estimate a six-degree-of-freedom position for the camera. The closed-loop tracker could be stabilized using Kalman filters, PID controllers or particle filters, all of which have been used previously for this kind of problem. Stabilizing a closed-loop solution would allow us to do more interesting things with the tracker, like dynamically

adjusting the markers positions and dimensions to make them optimal for both the camera and the audience.

The results of a controlled study comparing the camera pointing device to a mouse were reported, and they show that the mouse performs about twice as fast as the camera, while error rates are comparable between the devices. Our belief is that this is a good trade-off for classroom users, who often interact only infrequently with the display and will enjoy the increased usability of direct pointing. In the future we would like to work with lecturers that are willing to help us test the device in an actual classroom setting, to gather field data that will be very valuable in creating a better classroom interface. We would also like to run further studies where control to display ratio values are adjusted at different distances from the display, to measure the effects on device reliability and precision.

Many of our study participants mentioned the lag in the device was noticeable, but not problematic. Understanding the sources and effects of lag in the system would give us a better sense of how well the device can work with faster hardware. Our conjecture is that reduced system latency would increase user throughput. We would like to run a second study that looks into the effect of lag on user performance with the chiroptic tracker, which would allow us to estimate how much improvement in performance could be expected from reduced lag.

We presented ideas for the creation of an interface that makes use of a direct pointing device and a 5-button clicker to encourage the discussion of what a classroom interface should be like. Future work will implement these techniques and test their effectiveness with real users.

The prototype implementation for the chiroptic tracker that was presented here exceeded some of our initial expectations. It is possible that it can be used effectively in other settings than the one we designed it for. We would like to do more explorations of the physical design to create a technology that can be flexible enough to be used with a camera phone (which, while not the optimal form factor, might be a convenient way to have a chiroptic tracker readily available), but also embedded into clothing or other wearables. This flexibility would enable the creation of more sophisticated

interfaces with interactions that are similar to those found in multi-touch interfaces. In any case more effort should be put into the design of a chi-roptic tracker as a dedicated device, one that grants lecturers a robust set of interactions with which to create a rich classroom experience.

Bibliography

- [1] Vivek Agarwal, Besma R. Abidi, Andreas Koschan, and Mongi A. Abidi. An overview of color constancy algorithms. *Journal of Pattern Recognition Research*, 1(1):42–54, 2006.
- [2] Karl Johan Aström and Richard M. Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton University Press, 2010.
- [3] Matthias Baldauf, Peter Fröhlich, and Katrin Lasinger. A scalable framework for markerless camera-based smartphone interaction with large public displays. In *Proceedings of the 2012 International Symposium on Pervasive Displays*, PerDis '12, pages 4:1–4:5, New York, NY, USA, 2012. ACM.
- [4] Rafael Ballagas, Jan Borchers, Michael Rohs, and Jennifer G. Sheridan. The smart phone: a ubiquitous input device. *Pervasive Computing, IEEE*, 5(1):70–77, 2006.
- [5] Rafael Ballagas, Michael Rohs, and Jennifer G. Sheridan. Sweep and point and shoot: Phonecam-based interactions for large public displays. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1200–1203, New York, NY, USA, 2005. ACM.
- [6] Peter Beshai. Implementation and evaluation of a classroom synchronous participation system. Master's thesis, University of British Columbia, Vancouver, BC, Canada, 2014.
- [7] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
- [8] Bill Buxton. Some milestones in computer input devices: an informal timeline. <http://www.billbuxton.com/inputTimeline.html>. Accessed August 12, 2015.

- [9] Duncan Cavens, Florian Vogt, Sidney Fels, and Michael Meitner. Interacting with the big screen: Pointers to ponder. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '02, pages 678–679, New York, NY, USA, 2002. ACM.
- [10] Cesare Celozzi, Gianluca Paravati, Andrea Sanna, and Fabrizio Lamberti. A 6-dof ARTag-based tracking system. *Consumer Electronics, IEEE Transactions on*, 56(1):203–210, 2010.
- [11] Matthias Deller and Achim Ebert. ModControl - mobile phones as a versatile interaction device for large screen applications. In *Proceedings of the 13th IFIP TC 13 International Conference on Human-computer Interaction, Part II*, INTERACT '11, pages 289–296, Berlin, Heidelberg, 2011. Springer-Verlag.
- [12] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society.
- [13] Paul M Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 47(6):381–391, 1954.
- [14] David Fofi, Tadeusz Sliwa, and Yvon Voisin. A comparative survey on invisible structured light. In *Proc. SPIE*, volume 5303, pages 90–98. Machine Vision Applications in Industrial Inspection XII, 2004.
- [15] David A. Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall, 2003.
- [16] Orazio Gallo, Sonia M. Arteaga, and James E. Davis. Camera-based pointing interface for mobile devices. In *15th IEEE International Conference on Image Processing*, ICIP '08, pages 1420–1423. IEEE, 2008.
- [17] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Computational color constancy: Survey and experiments. *IEEE Transactions on Image Processing*, 20(9):2475–2489, 2011.

- [18] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, second edition, 2003.
- [19] NaturalPoint Inc. OptiTrack. <http://www.optitrack.com/>. Accessed August 17, 2015.
- [20] Seokhee Jeon, Jane Hwang, Gerard J. Kim, and Mark Billinghurst. Interaction techniques in large display environments using hand-held devices. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '06, pages 100–103, New York, NY, USA, 2006. ACM.
- [21] Hao Jiang, Eyal Ofek, Neema Moraveji, and Yuanchun Shi. Direct Pointer: Direct manipulation for large-display interaction using hand-held cameras. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 1107–1110, New York, NY, USA, 2006. ACM.
- [22] Carsten Kirstein and Heinrich Mueller. Interaction with a projection screen using a camera-tracked laser pointer. In *Proceedings of Multimedia Modeling*, MMM '98, pages 191–192. IEEE, 1998.
- [23] Zhangbo Liu. LACOME: a cross-platform multi-user collaboration system for a shared large display. Master's thesis, University of British Columbia, Vancouver, BC, Canada, 2007.
- [24] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. 7th Intl. Joint Conf. on Artificial Intelligence*, volume 81 of *IJCAI*, pages 674–679, 1981.
- [25] I. Scott MacKenzie. Movement time prediction in human-computer interfaces. In Ronald M. Baecker, Jonathan Grudin, William A. S. Buxton, and Saul Greenberg, editors, *Human-computer Interaction*, pages 483–492. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

- [26] I. Scott MacKenzie and Shaidah Jusoh. An evaluation of two input devices for remote pointing. In Murray R. Little and Laurence Nigay, editors, *Engineering for Human-Computer Interaction*, volume 2254 of *Lecture Notes in Computer Science*, pages 235–250. Springer Berlin Heidelberg, 2001.
- [27] I. Scott MacKenzie and Colin Ware. Lag as a determinant of human performance in interactive systems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 488–493, New York, NY, USA, 1993. ACM.
- [28] Russell MacKenzie. LACOME: Early evaluation and further development of a multi-user collaboration system for shared large displays. Master's thesis, University of British Columbia, Vancouver, BC, Canada, 2010.
- [29] Anil Madhavapeddy, David Scott, Richard Sharp, and Eben Upton. Using camera-phones to enhance human-computer interaction. In *Sixth International Conference on Ubiquitous Computing (Adjunct Proceedings: Demos)*, 2004.
- [30] Kento Miyaoku, Suguru Higashino, and Yoshinobu Tonomura. C-blink: A hue-difference-based light signal marker for large screen interaction via any mobile terminal. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, pages 147–156, New York, NY, USA, 2004. ACM.
- [31] Orkhan Muradov. Feasibility of supporting pointing on large wall displays using off-the-shelf consumer-grade tracking equipment. Master's thesis, University of British Columbia, Vancouver, BC, Canada, 2013.
- [32] Brad A. Myers, Rishi Bhatnagar, Jeffrey Nichols, Choon Hong Peck, Dave Kong, Robert Miller, and A. Chris Long. Interacting at a distance: Measuring the performance of laser pointers and other devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 33–40, New York, NY, USA, 2002. ACM.

- [33] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR 2011.
- [34] Donald A. Norman. *The psychology of everyday things*. Basic books, 1988.
- [35] Katsuhiko Ogata. *Modern Control Engineering*. Prentice Hall, fifth edition, 2010.
- [36] Dan R. Olsen Jr. and Travis Nielsen. Laser pointer interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 17–22, New York, NY, USA, 2001. ACM.
- [37] James G. Phillips and Thomas J. Triggs. Characteristics of cursor trajectories controlled by the computer mouse. *Ergonomics*, 44(5):527–536, 2001.
- [38] Barry A. Po, Brian D. Fisher, and Kellogg S. Booth. Comparing cursor orientations for mouse, pointer, and pen interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 291–300, New York, NY, USA, 2005. ACM.
- [39] Vasanth Kumar Rajendran. Interaction with large stereoscopic displays: Fitts and multiple object tracking studies for virtual reality. Master's thesis, University of British Columbia, Vancouver, BC, Canada, 2012.
- [40] Michael Rohs. Real-world interaction with camera phones. In *Proceedings of the Second International Conference on Ubiquitous Computing Systems*, UCS '04, pages 74–89, Berlin, Heidelberg, 2005. Springer-Verlag.
- [41] Joaquim Salvi, Jordi Pagès, and Joan Batlle. Pattern codification

- strategies in structured light systems. *Pattern Recognition*, 37(4):827–849, 2004.
- [42] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rehmann, Ido Leichter, Alon Vinnikov, Yichen Wei, Daniel Freedman, Pushmeet Kohli, Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3633–3642. ACM, 2015.
- [43] Junhao Shi. Improve classroom interaction and collaboration using i>Clicker. Master’s thesis, University of British Columbia, Vancouver, BC, Canada, 2013.
- [44] Garth Shoemaker, Takayuki Tsukitani, Yoshifumi Kitamura, and Kellogg S. Booth. Two-part models capture the impact of gain on pointing performance. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 19(4):28:1–28:34, December 2012.
- [45] R. William Soukoreff and I. Scott MacKenzie. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts’ Law research in HCI. *International Journal of Human-Computer Studies*, 61(6):751–789, December 2004.
- [46] Ivan E. Sutherland. Sketchpad: a man-machine graphical communication system. Technical Report 574, University of Cambridge, Computer Laboratory, September 2003.
- [47] Vicon Motion Systems. Vicon. <http://www.vicon.com/>. Accessed August 17, 2015.
- [48] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

- [49] Florian Vogt, Justin Wong, Sidney Fels, and Duncan Cavens. Tracking multiple laser pointers for large screen interaction. In *Extended Abstracts of ACM UIST*, pages 95–96, 2003.
- [50] Colin Ware. *Information Visualization: Perception for Design*. Elsevier, 2012.
- [51] Alan Traviss Welford. *Fundamentals of skill*. Methuen, 1968.

Appendix A

Experimental Resources

The following are the documents given to participants of the pointing study described in Chapter 4.

A.1 Consent Form

Participants were asked to read and sign this consent form.

A.1. Consent Form



Interacting With Large Displays V

UBC Department of Computer Science
ICICS/CS Building
201-2366 Main Mall
Vancouver, B.C., V6T 1Z4

Consent Form

Principal Investigator

Kellogg S. Booth, Professor, Department of Computer Science, [REDACTED]

Co-Investigator

Francisco Escalona, M.Sc. Student, Department of Computer Science, [REDACTED]
Benjamin F. Janzen, M.Sc. Student, Department of Computer Science, [REDACTED]

Project Purpose and Procedures

The purpose of this study is to evaluate different methods for interacting with different sized electronic displays. Because of the properties of some displays, and how they are used, standard devices such as mice and keyboards are ill suited. You will be asked to point at targets on the display, and press a button to select the targets.

Confidentiality

Your identity will remain anonymous and will be kept confidential. A computer will record performance and motion data as you perform the tasks, but no identifying information (such as your name) will be stored with this data, nor will it be associated with the data after it has been analyzed.

The results will be made public through publications; however, no identifying information will be included in any published disclosure of the research.

No audio recordings or photographs will be made of your participation.

Risks/Remuneration/Compensation

There are no anticipated risks to you participating in this research. Use of stereoscopic glasses may cause slight discomfort or fatigue in some subjects. You are free to take a break or withdraw from the study.

You will receive an honorarium of \$10 for your participation. You will be eligible for the honorarium even if you withdraw from the study.

A.1. Consent Form

Contact Information about the Project

If you have any questions or require further information about the project you may contact Francisco Escalona ([REDACTED] or [REDACTED]), Benjamin Janzen ([REDACTED] or [REDACTED]), or Dr. Kellogg Booth ([REDACTED] or [REDACTED]).

Contact for Concerns About the Rights of Research Subjects

If you have any concerns about your treatment or rights as a research participant and/or your experiences while participating in this study, contact the Research Participant Complaint Line in the UBC Office of Research Services at [REDACTED] or if long distance e-mail to [REDACTED] or call toll free [REDACTED].

Consent

We intend for your participation in this project to be pleasant and stress-free. Taking part in this study is entirely up to you. You have the right to refuse to participate in this study. If you decide to take part, you may choose to pull out of the study at any time without giving a reason and without any negative consequence.

Your signature below indicates that you have received a copy of this consent form for your own records.

Your signature indicates that you consent to participate in this study.

I, (print name) _____ agree to participate in the study as outlined above. My participation in this study is voluntary and I understand that I may withdraw at any time.

Participant Signature

Date

Printed Name of the Participant signing above

A.2 Initial Questionnaire

This questionnaire was given to participants at the beginning of the study, to gather basic demographics and check they met the study's requirements.

A.2. Initial Questionnaire

Interaction With Large Displays III Phase 1 Pre-Experiment Questionnaire

Participant #

1. How old are you?
 years
2. What is your gender? *(tick one)*
 - ☐ Male
 - ☐ Female
 - ☐ Other
3. How much time do you spend **per week** using a computer? *(tick one)*
 - ☐ Less than 1 hour
 - ☐ 1 to 3 hours
 - ☐ 4 to 8 hours
 - ☐ More than 8 hours
4. Do you normally wear glasses or contact lenses? *(tick one)*
 - ☐ Yes
 - If yes, what is your prescription?
 - ☐ I don't know
 - ☐ No

A.3 Final Questionnaire

This questionnaire was given to participants at the end of the study, and they were instructed to report only on their experience with the camera, to gather subjective and qualitative data.

A.3. Final Questionnaire

Interaction With Large Displays II Phase 1
Post-Experiment Questionnaire

Participant #

1. Overall, what was the level of difficulty of the task? (circle one number)

(Easy) 1 2 3 4 5 (Impossible)

2. Did you employ any particular strategy in completing the task? Please explain.

3. Please write any other comments you have regarding your experience with this task:
