Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Dominic LABERGE

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

Master of Computer Science

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Visual Tracking for Human Computer Interaction

E. Petriu

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

J.-F. Lapointe

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

N. Georganas

G. Roth

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

SIGNATURE

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

VISUAL TRACKING FOR HUMAN COMPUTER INTERACTION

By
Dominic Laberge

A thesis submitted to
the Faculty of Graduate and Postdoctoral Studies
in Partial fulfillment of
the requirements for the degree of

Master of Computer Science

The Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering (SITE)
University of Ottawa
Ottawa, Ontario, Canada

August 22^{nd} 2003

Your file Votre référence
ISBN: 0-612-90099-1
Our file Notre référence
ISBN: 0-612-90099-1

UNIVERSITY OF OTTAWA

Date: **August 22nd 2003**

Author: **Dominic Laberge**

Title: **Visual tracking for human computer interaction**

Department: **School of Information Technology and Engineering**

Degree: **M.Sc.** Convocation: **August** Year: **2003**

Permission is herewith granted to the University of Ottawa to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Author

To Julie.

Table of Contents

Table of Contents iv

List of Tables viii

List of Figures ix

List of Symbols xi

Abstract xiii

Acknowledgments xiv

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Open Problems . . . 3
- 1.3 Research purpose . . . 3
- 1.4 Comparison criteria . . . 4
  - 1.4.1 Latency . . . 4
  - 1.4.2 Accuracy . . . 5
  - 1.4.3 Resolution . . . 6
  - 1.4.4 Jitter . . . 6
  - 1.4.5 Update rate . . . 6
  - 1.4.6 Range . . . 7
  - 1.4.7 Susceptibility to the environment . . . 7
- 1.5 Contributions . . . 8
- 1.6 Thesis Overview . . . 8

2 Background and Related work 10
- 2.1 Virtual Reality . . . 10
  - 2.1.1 Display technology . . . 11
  - 2.1.2 Sub-classes of VR . . . 14

2.1.3 The Virtual Reality Workbench . . . 14
2.2 Human-Computer Interaction . . . 15
2.2.1 Pointing Devices . . . 17
2.2.2 HCI and Virtual Reality . . . 18
2.3 Tracking For Interactions . . . 19
2.3.1 Acoustic . . . 19
2.3.2 Inertial . . . 21
2.3.3 Electromagnetic . . . 21
2.3.4 Mechanical . . . 22
2.3.5 Optical . . . 24
2.4 Camera . . . 26
2.4.1 Mathematical Model . . . 26
2.5 Calibration . . . 27
2.5.1 Manual calibration . . . 29
2.5.2 Auto-calibration . . . 30

3 Tracking through images **33**
3.1 Processing the input signal . . . 33
3.2 Pattern recognition . . . 34
3.2.1 Rigid vs. deformable pattern . . . 34
3.2.2 Natural vs. artificial features . . . 35
3.2.3 Template matching . . . 36
3.2.4 Corner and edge detection . . . 36
3.2.5 Blob detection . . . 36
3.2.6 Red-Blue detection . . . 37
3.3 Speed optimization . . . 39
3.4 Accuracy of the detected pattern position . . . 41
3.5 Experimental setup . . . 41

4 Reaching into VR: 4-D Interaction **42**
4.1 Device definition . . . 42
4.2 Targeted applications . . . 45
4.3 Experiment definition . . . 46
4.4 Performance results . . . 48
4.5 Analysis . . . 48
4.5.1 Range . . . 48
4.5.2 Accuracy . . . 49
4.5.3 Update rate . . . 49
4.5.4 Jitter . . . 49
4.5.5 Latency . . . 49
4.5.6 Infrared vs. Visible Pattern . . . 49

4.6 Discussion & future work 50

5 Full pose control: 6-D Mouse **52**
5.1 Device definition 52
5.1.1 Pose estimation 54
5.2 Targeted applications 58
5.3 Experiment definition 58
5.4 Performance results 58
5.5 Analysis 59
5.5.1 Range 59
5.5.2 Accuracy 59
5.5.3 Update rate 61
5.5.4 Jitter 61
5.5.5 Latency 62
5.5.6 Infrared vs. Visible Light LEDs 62
5.5.7 LEDs vs Micro lamps 62
5.6 Discussion & future work 63

6 2-D in 3-D: Laser Pointer **65**
6.1 Introduction 65
6.1.1 Previous work 66
6.2 Laser pointing algorithm 68
6.2.1 Assumptions 68
6.2.2 System setup 68
6.2.3 Preprocessing 69
6.2.4 Processing algorithm 70
6.3 Targeted applications 71
6.4 Experiment definition 71
6.5 Performance results 72
6.6 Analysis 73
6.6.1 Range 73
6.6.2 Accuracy 74
6.6.3 Update rate 76
6.6.4 Jitter 76
6.6.5 Latency 76
6.6.6 Infrared vs. visible laser pointer 76
6.7 Discussion & future work 77

7 Conclusion **79**

A Vision Graph Framework **82**
A.1 Active vs. Passive nodes 83
A.2 The node manager 83
A.3 Main nodes 83
A.3.1 Grabber nodes 83
A.3.2 Writer nodes 84
A.3.3 Processor nodes 85
A.4 Buffer Nodes 85
A.4.1 Queue buffer 86
A.4.2 Quick buffer 86
A.5 Other nodes 87
A.5.1 Synchronizer node 87
A.6 Examples of application 88

Bibliography **90**

List of Tables

2.1 Performance of a typical ultrasonic tracker (Logitech). [5] 20
2.2 Performance of a typical inertial tracker. [4, 52, 33] 21
2.3 Performance of a typical electromagnetic tracker. [7, 33] 23
2.4 Performance of a typical mechanical tracker. [2, 3] 23

3.1 Experimental setup 41

4.1 4-D mouse performance results 48

5.1 6-D mouse performance results 59

6.1 Laser pointer performance results 73

List of Figures

1.1 Standard referential definition. 5

2.1 Anthropocentric model of VR 12
2.2 The Virtual Reality Workbench 15
2.3 Parent fields of HCI. 16
2.4 Taxonomy of VR navigation tasks [17] 18
2.5 Taxonomy of VR manipulation/selection tasks [16] 19
2.6 Acoustic trackers 20
2.7 Ascension Magnetic Tracker 22
2.8 Camera coordinate systems [36] 28
2.9 Radial distortion [36] 29
2.10 Example of a calibration pattern 30

3.1 Four- and eight-connected blob detection. 37
3.2 Red-Blue detection algorithm. 38
3.3 Target centroid detection. 38
3.4 Recursive blob detection vs. Red-Blue detection 39
3.5 Detection range restriction 40

4.1 The 4-D Mouse 42
4.2 4-D Mouse system 43
4.3 4-D Mouse processing 44
4.4 4-D Mouse angle computation 45

5.1 The 6-D mouse prototypes: 1(left) and 2(right) 52

5.2 the 6-D Mouse pattern 53
5.3 Error along X axis for the 6DOF mouse 60

6.1 The laser pointer interface 65
6.2 The laser pointing system. 69

A.1 The Grabber node. 84
A.2 The Writer node. 84
A.3 The Processor node. 85
A.4 The Buffer node. 86
A.5 The QuickBuffer node. 87
A.6 Trivial VG application 88
A.7 Typical VG application 88

List of Symbols

c_1, c_2, c_3 : Cosine of angles θ_1, θ_2, θ_3

c_x, c_y : Pixel coordinate on the camera plane

DOF : Degree of Freedom

f_u : Focal length along the X axis

f_v : Focal length along the Y axis

f_k : Focal length of the camera (Camera coordinate system)

FOV : Camera's *Field-of-view* angle

H 3×3 : Homography matrix

$h_{11}...h_{33}$: Elements of the homography matrix

h, v : Image horizontal and vertical measurements of the viewable plane at a distance from the camera

λ (***lambda***) : Homography factor taking into account tangential distortion and distance to the projection plane

$L_1...L_5$: LED number on the 6-D Mouse pattern

L_t : LED assigned to the trigger on the 6-D Mouse pattern

s_1, s_2, s_3 : Sine of angles θ_1, θ_2, θ_3

s_x, s_y : Pixel size along both axis on the camera plane

t : Point in time

T_x, T_y, T_z : Transformation between the world

θ : Camera's *Field-of-view* angle

θ_1, θ_2, θ_3 : Rotation angles along X, Y and Z

x_i, y_i : Target coordinates in pattern space

x_i', y_i' : Target coordinates in the world coordinate system

x, y, z : 3D Coordinates

x, y, w : 2D Homogenous coordinates

X_k, Y_k, Z_k : Camera coordinate system

X_w, Y_w, Z_w : World coordinate system

z : Distance from the camera coordinate system and the camera coordinate system.

Abstract

The purpose of this master's thesis project is to design, implement and evaluate vision-based user interfaces for use in the context of virtual environments. Three interfaces are treated.

The first one is a 4 degrees of freedom (DOF) mouse that can track the position and 1 DOF or rotation (roll) of a user's hand.

The second one is a 6 DOF mouse that can track both the position and the orientation of a user's hand in 3D space.

Finally the third one is a laser pointing interface used to track the laser spot of a standard laser pointer, in order to interact with a large screen display.

The two latter interfaces use an auto-calibrated approach based on planar homography, that distinguish them from the standard computer-vision based approach which requires a previous step of calibration.

Acknowledgments

First, let me thank my current supervisors Dr. Emil Petriu and Dr. Jean-François Lapointe for their non-stop support, both moral and academic, throughout this masters degree.

Then I would like to thank my supervisor for the first year and a half Dr. Pierre Boulanger for giving me the opportunity I had in working jointly with the National Research Council and the University of Ottawa.

Thanks to the National Research Council, especially to Guy Godin, Louis Borgeat and Philippe Massicotte of the Visual Information Technology group and Dr. Gerhard Roth, Shahzad Malik and Dr. Mark Fiala of the Computational Video group for their help and constructive comments on my research.

For introducing me to the world of virtual reality, I definitely have to thank Daniel Lee Howe, a student who put time and effort into a project to provide a VR course at the undergraduate level. A course that was given in my last term as an undergraduate and which eventually led to my continuation to the masters level.

A special thank you to Pierre-Alexandre Fortin and Stéphane Lalande for allowing me to take pictures of them operating my input devices for my paper and for their help putting my mind off work and on frisbee during lunch hours.

Another special thank you to Winston Wong for helping me with the theoretical basis for the Vision Graph and for being a source of inspiration over the last two years.

Thanks to my friends Nicolas, Dominique, Jonathan and all those I do not have the space to name and especially to my fiancée Julie for supporting, encouraging and enduring me over the course of this thesis.

Presenting ongoing research in front of a group often helps in thinking of the key issues and working out communication skills. For giving me that opportunity I have to thank Dr. Nicolas Georganas of the Discover Lab who gave me that opportunity several times.

Finally, I have to thank the National Capital Institute for Telecommunication for their funding throughout this two years research project.

Chapter 1

Introduction

As those who dreamed of making a computer fit in a standard house, we now dream of making it fit in a human cell. And as those who dreamed of adding two integers within a second, we dream of making it as fast as an atom splits. Ubiquity of computing is becoming obvious. It is in our transportation, our industries, our relationships among each other. We are no longer defined by our name and our place of birth but by our internet nicknames, e-mail or IP addresses. The world is getting smaller, on the verge of the next great step in human collaboration.

Like many scientific achievements, virtual environments made their appearance in science fiction as an ethereal space where Man can interface his senses with the Machine. Movies like Johnny Mnemonic and the Matrix are good examples where someone is engulfed with or against its will in the digital world.

Now as for three of those senses, namely sight, hearing and touch, this is becoming science fact, smell is following not being far behind, yet one major problem is to be solved. How can we as humans interface with the computer. How can we clearly send our wishes to the machine without having to think too much about the way to do it? The best way overall

would be to be neurally connected to the machine like the latest science fiction writers seem to foresee. Some progress in that sense has already been made. But to get it working to a level of everyday life for everybody is not for today, probably not even for tomorrow.

In the meantime, we need tools, both software and hardware, to close the gap between Humans and Machines.

1.1 Motivation

Since virtual reality as a field is present in computer science and engineering, research has thrived to produce interfaces for users to interact with this new tool of mankind. Potential for applications is almost infinite but it is capped by the lack of proper interfaces.

Many years later, interfacing a human and a virtual environment is not a solved problem. That is what motivated research that spawned this thesis. In the view of the author of this work, it is imperative to solve the problem of interaction in order to tap into the full potential of virtual reality (VR).

As technology becomes more and more available to the masses, a need arises for interfaces that are efficient, yet simple to operate with minimal training, as well as for solutions that minimize the cost of the manufacturing process. Most interfaces to date are expensive and not trivial to set up and operate. This is a good motivation to create interfaces that use already existent "off the shelf" components or low-cost mass-producible interfaces.

1.2 Open Problems

The ultimate goal when interacting with a virtual environment is to have a low-cost device with a method that provides performance in human-machine interaction without encumbrance, that is totally transparent to the user without requiring as much computing power as to introduce lag.

One step towards that direction is to study what is available now in terms of technology, what devices can be designed using that technology, what are their performances in terms of speed and quality of data. When that step has been taken, improvements will be based on a solid scientific base.

A promising area of the technology is vision-based interfaces because of advances in processor speed and the dropping cost of reasonable quality cameras. To make the step mentioned above with these, is the open problem in the scope of this thesis.

1.3 Research purpose

The purpose of this research is divided into two sub-goals:

The first goal is to build interfaces that are adapted to use in the context of virtual reality applications. This means that different degrees of freedom (DOF) in interfaces are needed because of the diversity of the tasks. These interfaces have to use the computer vision technology available to solve the problem while remaining relatively inexpensive to produce. Concretely, this translates to:

- A 4 DOF mouse built on the principle of perspective projection.

- A 6 DOF mouse elaborated starting from a design of Kumar [37].
- A laser pointing interface for screen interaction.

The second goal is to evaluate the performances of those devices and from there, assess whether or not they would be useful by comparing their performances to other devices.

1.4 Comparison criteria

For a comparative study between multiple input devices to be useful, the criteria on which they are evaluated must be the same. That is why a set of criteria was chosen for this study. They are the following:

1.4.1 Latency

Latency, also called total lag, reaction time, overall system latency and system responsiveness, is defined as the elapsed time until the system responds to a user command [7, 60]. Part of it is directly related to signal processing, while input to output device latency and communication time make up for the rest. In our experiments, latency is measured by using a high speed camera (250 frames/s) with a temporal accuracy of $\pm 4ms$.

To take measurements, the input device is linked to an external event, in this case, the mouse cursor. The cursor is placed at an arbitrary position on the screen, then the inactive input device is placed on a stable surface. A video is recorded of the input device being activated and the projected screen shown on the same image. Then the number of frames separating the input device pattern, or laser dot, being activated and the first movement in the cursor is counted manually. That number is then multiplied by 0.04 s (1 / camera

frame rate) to give an estimate of latency.

1.4.2 Accuracy

In the case of pose estimation, accuracy is defined by the difference between the real pose and the one computed by the algorithm. It can be divided into two subgroups, namely: positional and angular respectively comparing x, y, and z coordinates or *pitch*, *roll* and *yaw* angles.

For two out of three experiments presented here, the ones that involve three dimensional interaction, accuracy is calculated using a measurement table built for that purpose. It is equipped with a gimbal that can slide along the three axes, x, y and z, aligned with the camera axes. It can also be rotated around each of those axes, thus causing heading, pitch and roll movements. The accuracy of the table is limited by the rulers used to take measurement, in our case $\pm 0.5mm$ and $\pm 0.5^o$.

The referential used to measure pose in our experiments is the following:

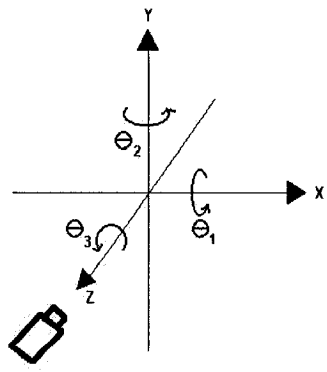

Figure 1.1: Standard referential definition.

To take position accuracy measurements, 25 measurements are taken uniformly on a plane with a fixed z component, parallel to the camera plane. This operation is then repeated two times at different z values. The measurements given by the algorithm are compared to the measurements taken on the measurement table. The worst position error, either positive or negative, is taken as a higher bound and reported in this thesis.

To take orientation accuracy measurements, an angle is set on the platform under the input device on the measurement table and tightened in place. Angle measurements given by the algorithm are then recorded for 25 different positions on a plane with fixed z component parallel to the camera plane. The operation is repeated two times at different z values. The worst angle error, either positive or negative, is taken as a higher bound and reported.

1.4.3 Resolution

Resolution is defined as the smallest movement that can be detected by the tracking system. We measure both spatial and angular resolution.

1.4.4 Jitter

Jitter is a measure of stability. It is measured when tracking an immobile pattern. Differences over time in detection is what cause jitter. Jitter measures are rarely given in units, they are rather given qualitatively.

1.4.5 Update rate

The update rate of the tracking interface is measured in Hertz(Hz). A higher update rate normally results in smoother tracking like better screen resolution results in smoother images. Two update rates are measured. The first one uses the algorithm's execution time

at each frame. This is the maximum update rate assuming no other delays.

The second one is calculated with a high speed camera. A video of the user's screen is recorded as the input device is linked to the mouse cursor. The number of frames between each position update is counted manually. Since the camera is set to 250 frames/second, the algorithm runs at $\frac{250\ Hz}{number\ of\ frames\ observed\ between\ each\ movements}$.

The latency is ultimately limited by the update rate, but other factors such as transmission time can affect it.

1.4.6 Range

The range of an input device defines the volume a user can work in, with the device functioning properly. It has a minimum and a maximum value. Some devices that do not need an external transmitter or receiver are said to be relative. They have no anchor point and therefore have an infinite range.

1.4.7 Susceptibility to the environment

The environment can and usually has an effect on tracking performance. In the case of optical tracking, the lighting is probably the most potent disturbing element.

Optical tracking devices generally perform better when the contrast between the tracked pattern and the surrounding environment is maximized. Computing under different light conditions around the camera will allow us to measure the impact of surrounding light on the tracking performance with the surrounding.

Usually, in the context of immersive virtual reality rooms such as the Virtual Reality Workbench (See section 2.1.3), the room is relatively dark to increase the feeling of presence

of the subject. That is why active, i.e. light emitting, patterns are most of the time required to provide the necessary contrast with the surroundings.

1.5 Contributions

Contributions that originated from the work of this thesis are on several levels.

First, we built and evaluated the tracking performances of several pose trackers. Two of those trackers were prototypes of a six DOF mouse where the second was an improved, more versatile version of the first one with interchangeable patterns. Also, a prototype of a four degrees of freedom tracker was built for experimentation purposes.

At the software level, a parallel software architecture, called *Vision Graph*, was created to support the experiments. Its architecture and implementation is explained in Annex A.

Finally, two paper were published from this thesis work [39, 38] and others are in progress.

1.6 Thesis Overview

Here is an overview of the content of this thesis:

This chapter introduces the subject of the thesis.

Chapter 2 reviews technology and concepts to provide the mathematical and theoretical background necessary to the understanding of the work.

Chapter 3 presents the concept of tracking through a sequence of images, as well as the target detection and tracking algorithm used for the rest of the work.

Chapter 4 describes and studies the performance of a four degrees of freedom mouse designed for VR applications.

Chapter 5 presents and analyzes the performance of our second and final prototype of six degrees of freedom mouse.

Chapter 6 describes a laser pointing interface, used in the context of virtual environments. A study of its performance is presented and discussed.

Chapter 7 concludes by an overall summary and discussion on the results presented in this thesis with pointers on the work that needs to be pursued in view of the current findings.

Chapter 2

Background and Related work

2.1 Virtual Reality

Virtual Reality, also known as Virtual Environments (VE), consists of artificially created environments that allow interaction with one or many users. It is a field of computer science that originates from three main fields: Computer Graphics, Simulation and Teleoperation.

The expression Virtual Reality (VR) has been around since the end of the 80s when Jaron Lanier coined it as the embodiment of an interactive simulation of sensorial input generated by a computer. A rather complete definition of VR is, as Fuchs & al propose in [26], *the interaction in realtime with a virtual environment using behavioral interfaces permitting pseudo-natural immersion of the user in this environment.*

This compact definition requires some explanation. *Realtime interaction* means that the user must not be able to perceive the time taken by the computer to react to its inputs. This delay, called *latency*, if too large, can reduce the interaction and break immersion.

Pseudo-natural immersion, also called *Presence*, is the concept that ties the knot around all other concepts and fields of VR. It is the principle of belief that makes a VR application

reach its goal, totally transparent user interaction. Even though the user himself knows that s/he is in a room with computers, s/he will unintentionally go beyond this and believe in the virtual environment as a fact [53]. Humans have learned to interact with the environment, the real one they live in. Immersion is the key for extending this behavior to a synthetic one *where different tasks defined by computer programs can be achieved in realtime with human interaction requiring minimal acquired behaviors* [26]. It is important to note here that imagination is not required from the subject to achieve immersion. At some point the brain just stops processing disbelief and gives in to immersion.

Behavioral interfaces define an abstraction of hardware and software input and output interfaces used in VR following an anthropocentric model (Figure 2.1)[26]. It is divided in two sections: *Sensorial Interfaces* and *Motor Interfaces. Sensorial Interfaces* transmit information from the computer to the user using a variety of ways to stimulate the appropriate senses. Those outputs lead to different levels of immersion based on the level of saturation of the senses they provide. The scope of this thesis leads to the second type of interfaces namely the *Motor Interfaces.* They are an abstraction of the users' tools for interaction with the Virtual Environment. But first, to illustrate different levels of immersion and set the frame for interaction, here is a diagram describing a VR interface.

2.1.1 Display technology

Like the way visual feedback sent to the user varies from one application to another, so does the immersion level. Immersion does not necessarily involve the user being completely encompassed in the virtual environment.

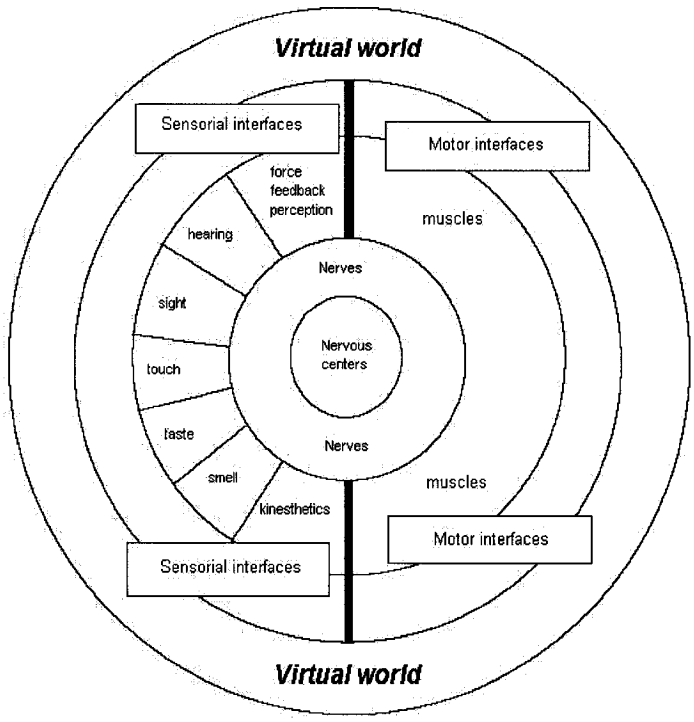

Figure 2.1: Anthropocentric model of VR

The first level of immersion, the least immersive, is at the desktop level. This is the level where the user only has a window to the virtual environment. It is often called *Fishtank VR* as an analogy to the marine attraction, or simple fish tank at home, where onlookers see the marine world through a glass without being "immersed" in it. The principle of belief resides in the certainty that there is really something behind that glass. Images can be projected in three dimensions create an *out of the screen* experience but the user is still constrained to his chair in front of the computer screen.

At a higher level is the *Desk VR*, where a desk or table is the playground and objects

are projected on or over that surface for the users to interact with. It is different from single screen VR in the fact that user can usually go around like they would do around a mock-up. At this level, the user is still not immersed *in* the environment but s/he is no longer interacting through a window. The desk is a protrusion of the virtual world into the real one, kind of like a *Twilight Zone.*

From this point on, immersion flips side. The user is no longer out in the real world but rather *in* the virtual environment. S/He will no longer only think that there are objects there, s/he will believe that they share the same environment. Large screen VR provides that type of immersion, if attention of the user is not drawn away by other real world elements.

More complex systems fill more the field of view of the user by adding screens and modifying geometry up to a point where no matter where the user turns there will be a view port into the virtual world.

Opaque head mounted display (HMD) technology is an alternative that requires less room to operate than projectors and screens, but also has its limitations. Wiring needed to link the video signal at acceptable speed is cumbersome, weight is also a factor in encumbrance. Lag in tracking the HMD directly results in latency in the display of the correct scene in front of the user's eyes. Current HMDs often give the impression the user is looking through a periscope because of a small *field of view.*

2.1.2 Sub-classes of VR

VR as a whole presents a number of sub-branches. To illustrate this branching, at the same time showing diversity of the field, here are two examples of sub-branches: Virtualized Reality and Augmented Reality.

Virtualized Reality (VzR) is a term coined by Guy Godin in 1994 [9]. The main concept behind VzR is to simulate real worlds in virtual environment to take advantage of the interaction techniques that are inherent to VR. It differs from traditional VR by the fact that virtual objects are not modeled from scratch but rather are digitized directly from real objects with the help of sensors. Proofs of concept of the use of VzR were conducted in different application fields such as : heritage [14] and teleoperation [13]. In the heritage field for example, it was successfully demonstrated that artifacts can be preserved and manipulated by using virtualized reality. In teleoperation, it was demonstrated that this technology can be used to remotely monitor a large area and control automated vehicles in realtime.

Augmented Reality Augmented reality (AR) is a concept that enables superposition of synthetic data over real world images [8, 41] through the use of different display devices.

2.1.3 The Virtual Reality Workbench

The Virtual Reality Workbench, as imagined by Dr. Pierre Boulanger and colleagues from the National Research Council, is an enclosed space, generally known as immersive room, where one or many users are in front of one or many screens which provide a view of the

virtual world. The work is divided on multiple machines to enhance performance.

Multiple servers share the task of analyzing and sending data to a central server that manages the audio and visual rendering of the scene.

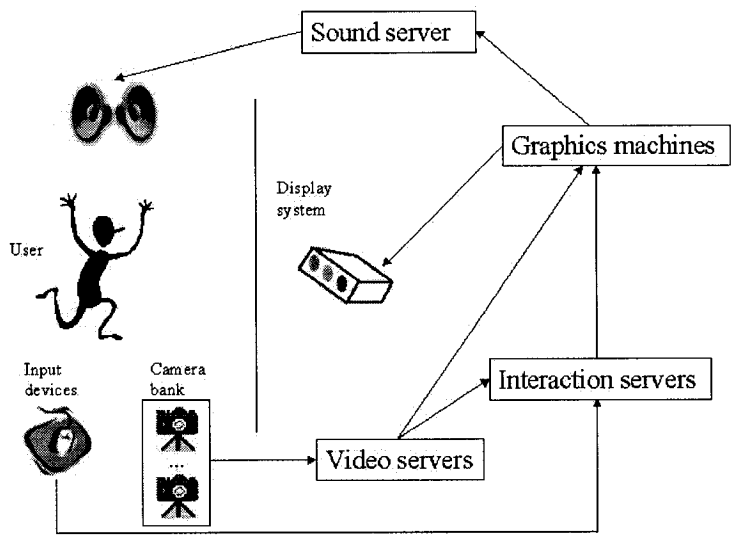

Figure 2.2: The Virtual Reality Workbench

2.2 Human-Computer Interaction

Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.

So are the words that are used to describe the field of human-computer interaction (HCI)[29]. A research subject that stretches from human to machine, from the computer's software to human perceptions and actions, bridging two information processing entities,

one biological, one artificial.

This field is born out of the symbiosis of mainly six research topics (figure 2.3) combining knowledge that has been accumulated mainly over the last century.[29]

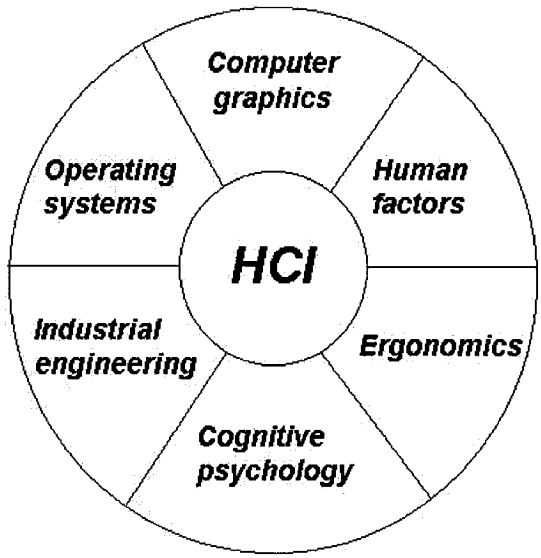

Figure 2.3: Parent fields of HCI.

In computer science, it often comes down to the design of efficient graphical user interfaces(GUI). Basically, this means to make use of available devices in creating or optimizing interaction techniques to solve known problems.

More on the engineering side, HCI thrives to design and develop devices that will better fill the gap between human and machine.

The field as a whole serves as a guide for application driven development, providing guidelines, data and models to improve the design of interface for particular tasks and predict human performance in human driven systems.

2.2.1 Pointing Devices

Of all the devices available as a motor interface for interacting with a computer, part of this thesis focuses on a specific, yet wide, class known as pointing devices. The name comes from the action the user accomplishes using those devices, namely pointing at things in a $N - dimensional$ space.

Tasks accomplished by pointing devices were defined and standardized in [25] and revisited in [54]. They are as follows:

Select Choose from a set.

Position Move, create or drag an object in N dimensions.

Orient Choose a direction with N degrees of freedom.

Path Create an itinerary through checkpoints.

Quantify Select a numerical value. Usually in $1 - D$.

Text Move, edit, enter text in $2 - D$.

Pointing devices can provide direct-control or indirect-control. Direct-control pointing devices provide interaction directly with the work surface. For example, the now rarely used *lightpen* worked by direct contact with the screen with occlusion problems that this creates. Indirect-control devices such as the computer *mouse* defer pointing to a secondary space which is correlated to the computer space.

The task of selection has perhaps been the most studied until now and a mathematical model has been created to quantify the time humans take to select targets [24].

2.2.2 HCI and Virtual Reality

By its interactive nature, virtual reality challenges HCI researchers to go forward and define, improve and innovate to increase the usability of computer systems.

The tasks that can be accomplished in virtual environments have been studied by Bowman & al. for navigation in [17] and for selection/manipulation in [16] (Figures 2.4 and 2.5).

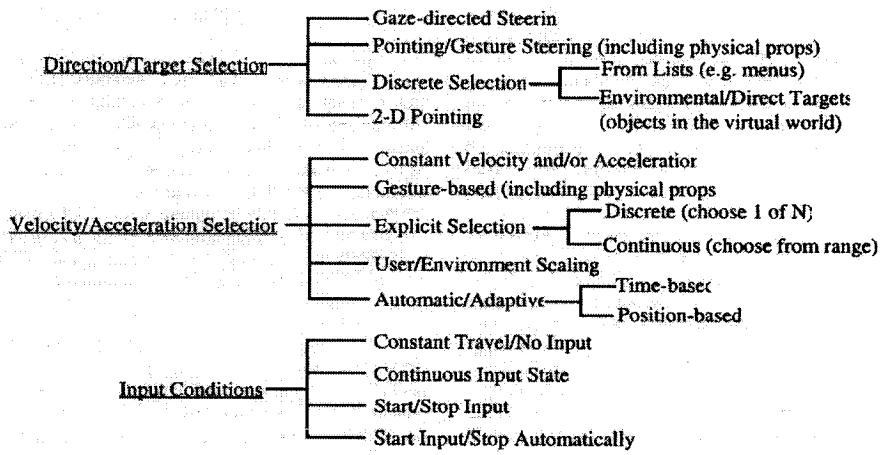

Figure 2.4: Taxonomy of VR navigation tasks [17]

Others focus on making tasks easier to the user, for example by evaluating six degrees of freedom docking techniques [12], or evaluation of specific tasks and optimization possibilities [40, 56].

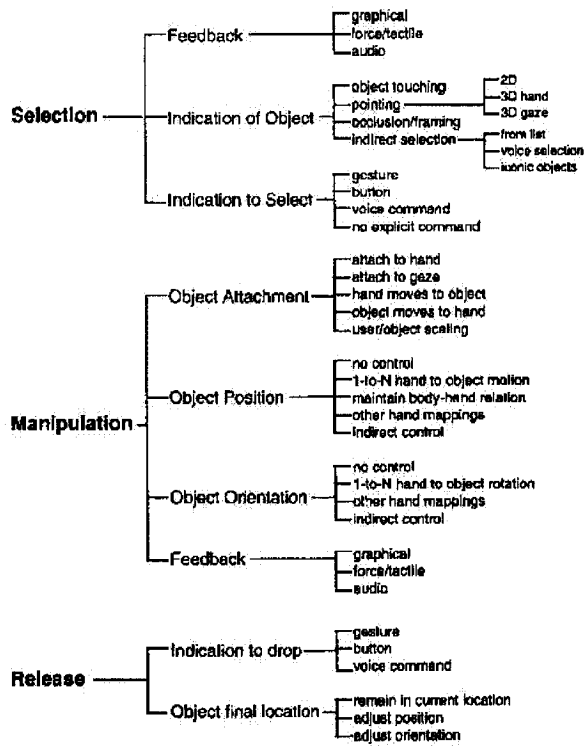

Figure 2.5: Taxonomy of VR manipulation/selection tasks [16]

2.3 Tracking For Interactions

One important aspect of HCI is to track the position and orientation of specific points of the environment. Trackers can use several different techniques such as: acoustic, inertial, electromagnetic, mechanical or optical. Performance of the trackers is evaluated with the criteria listed in section 1.4.

2.3.1 Acoustic

Acoustic tracking technology uses sound to find the pose of an object in space. Usually, the tracker is composed of a mobile emitter, itself composed of one or many ultrasound

sources placed in a known configuration, and of a stationary receiver, composed of many microphones (usually three) again placed in a known configuration. The pose of the mobile emitter is triangulated using time of flight calculations between emitters and microphones.

Logitech offers a commercial ultrasound tracking device based on these principles. Absolute three-dimensional positioning data is extracted by using three emitters and three microphones. Performance of their tracker is given in table 2.1. The *Mattel PowerGlove* has been another popular acoustic tracker targeted at user-end console gaming.

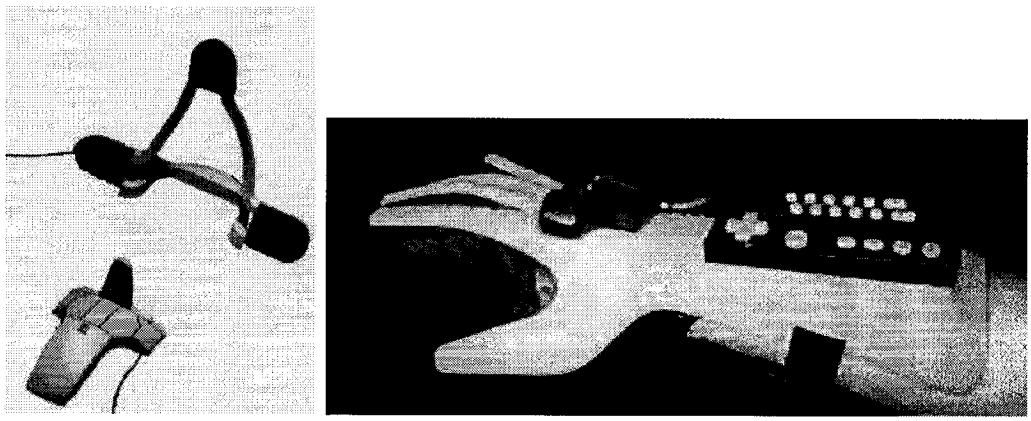

Figure 2.6: Acoustic trackers

Table 2.1: Performance of a typical ultrasonic tracker (Logitech). [5]

Criteria	Value
Latency	30 ms
Range	1.83 m, 100^o cone
Resolution (Position)	(2D) 0.06 mm (3D) 0.1 mm
Resolution (Orientation)	0.1^o
Update rate	1000 Hz

2.3.2 Inertial

Inertial trackers use the principle of inertia of mass to compute their position and orientation. Those trackers are of two types.

The first type uses optical gyroscopes to get position and orientation. Like all gyroscopes, they constantly drift, meaning that at intervals, the position of the gyro has to be updated by a second method. This method is however precise in orientation measurement.

The second type of inertial trackers translational uses acceleration sensors to sense movement and update position and orientation by integrating the acceleration over time. This method is often combined with the previous one to create a 6 DOF tracker.

Table 2.2: Performance of a typical inertial tracker. [4, 52, 33]

Criteria	Value
Latency	2 ms to 38 ms
Range (Position)	± 4 mm
Range (Orientation)	Heading/Roll $\pm 180^o$, Pitch $\pm 90^o$
Resolution (Position)	1.5 mm
Resolution (Orientation)	0.05^o
Accuracy (Position)	3 mm to 4 mm
Accuracy (Orientation)	Pitch/Roll: 0.1^o to 0.2^o, Heading: 0.15^o to 0.4^o
Update rate	250 Hz

2.3.3 Electromagnetic

Electromagnetic trackers use electromagnetic fields emitted from three orthogonal coils in a transmitter. The fields create electrical activity current in three other orthogonal coils located in the receptor. By analyzing the signals then created, position and orientation of the receptor can be computed [49].

This method is perhaps one of the earliest forms of 6DOF tracking. It is prone to interference due to nearby ferromagnetic materials since it assumes uniformity of the magnetic field. One way to compensate is to calibrate the device[30, 34]. This leads to better results but increases latency.

Magnetic devices until now have all been cumbersome either because the user is wired to the system or because he is wearing wireless data transmitting equipment. Also latency increases with the number of trackers.

Two commercial companies own the vast majority of the electromagnetic tracker market, namely, *Ascension technology*[1] and *Polhemus*[6].

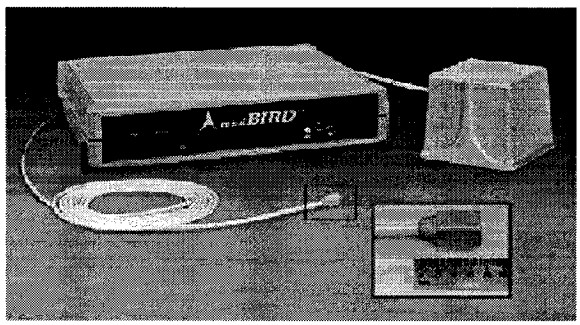

Figure 2.7: Ascension Magnetic Tracker

Let us just cite here another class of magnetic trackers that uses a compass to provide one degree of freedom always pointing to the magnetic north pole.

2.3.4 Mechanical

Mechanical tracking is one of the most reliable and fast tracking methods since the beginning of VR. It mainly makes use of electro-mechanical or optical sensors to measure with

Table 2.3: Performance of a typical electromagnetic tracker. [7, 33]

Criteria	Value
Latency	4 ms
Range	0.92 m(Ascension Flock of Bird) to 4.57 m(Polhemus UltraTrack)
Resolution (Position)	0.5 mm at 30.5 cm (Ascension Flock of Bird)
Resolution (Orientation)	0.1^o at 30.5 cm (Ascension Flock of Bird)
Accuracy (Position)	0.76 mm(Polhemus FasTrack) to 76 mm(Polhemus UltraTrack)
Accuracy (Orientation)	0.15^o(Polhemus FasTrack) to 3^o(Polhemus UltraTrack)
Update rate	Up to 144 Hz

precision, pose of a mechanical arm joint.

There are large and small scale trackers. For large scale tracking, for example, tracking a user's movements, a boom is often used. It is a set of rods with rotation sensors at the joints. A direct kinematics transform is used to compute position of the end effector of the boom which coincides with the user's tracked body part.

This technology require the user to be attached and restrained by it in order to be tracked, it is therefore not transparent nor is it comfortable in most cases.

Although mechanical trackers are well suited for haptic systems, due to their inherent input/output capabilities, they are cumbersome and often require calibration before being used.

Table 2.4: Performance of a typical mechanical tracker. [2, 3]

Criteria	Value
Latency	200 ns
Range	Typically around 1 m^3
Accuracy (Position)	0.005 mm (FaroArm Platinum) to 4.06 mm(Fakespace Boom)
Accuracy (Orientation)	0.1^o (Fakespace Boom)
Update rate	70 Hz

2.3.5 Optical

Optical tracking can be divided in two areas depending on the type of images used, namely *range* or *intensity*.

In range images, each pixel has three to six coordinates. X, Y, and Z coordinates in every case, added to that can be intensity information or red green and blue (RGB) coordinates in color space. This is explained for example in [11].

The second type, intensity images, is the standard two dimensional image produced by cameras. Every pixel has a coordinate (U, V) which locates it in terms of row and column. In black and white images, each pixel has one intensity value that can be sampled on 1 to n bits. A color image on the other hand has three intensity values, like the color range finder, one in red, green and blue.

Tracking with intensity images has been studied for many years in the field of computer vision. A multitude of tracking algorithms were developed. Of those some are tracking the human body. A few specialize in tracking specific facial features such as the nose [27], many track the hand [50, 51, 61] or the face as a whole[10, 31, 46, 59]. Those kinds of trackers focus on some parts of the human body , or features, *most* have in common. This is a problem in a HCI perspective, because one of the founding principles of HCI is to recognize human diversity in order to increase accessibility [54]. All humans are different and if accuracy is a goal, slight differences or absence of specific features can cause the tracking to fail.

The alternative is to remove the degree of freedom related to human diversity and use an input device. This interaction medium is less if at all tied to human features. For example,

disabled people with no arms can still use his feet to operate a computer mouse where a hand tracker would be of absolutely no use.

Because of the diversity of algorithms and since it is the scope of this thesis, a more complete analysis of the performance criteria will be given.

Latency in optical tracking systems varies from one algorithm to another and is largely related to processing time. It is therefore not appropriate to state a general value, since it is related with processing power of the computer. Given enough processing power, latency will be close to the image acquisition period of the camera.

Range is also special for optical tracking. First, the camera must be able to maintain a *line of sight* with the tracked device. If it is occluded and no prediction is involved, tracking will fail. The second range factor is in the intensity of light coming into the camera, if the tracked object does not reflect or emit sufficient light for its features to be detected by contrast with the environment, tracking will fail. This usually happens when the tracked subject is too far from the camera or ambient light conditions are not appropriate.

Accuracy depends on camera resolution and on the quality of the algorithm.

Resolution is a topic of interest with optical devices because it varies linearly relative to the distance from the tracked subject to the camera.

Update rate depends on the algorithm and processing speed but is ultimately limited by the acquisition rate of the camera.

2.4 Camera

Computer vision works essentially in the same way as human vision. On the human side, light coming from a source is reflected on objects and the reflected light carrying different intensity information is then transmitted to our eyes which act as sensors. Light at different wavelengths activate different light sensors on our retina. Narrow bandwidth sensors, the cones, are sensitive to specific colors: red, green and blue. Wide bandwidth sensors, the rods, only perceive the intensity of the light. The brain integrates this information to interpret the image.

On the computer side, light activates cells on the retinal plane of the camera, which can sense either general light intensity or more specific wavelengths to give color images. This retinal plane is composed of photosensitive electronic cells.

The camera is the essential part of vision and one of its greatest challenges. In its simplest theoretical expression, it is called a "Pinhole camera". This device is defined as a mathematical entity that performs perspective projection on an image point Π perpendicular to an optical axis centered on the focal point C.[57].

2.4.1 Mathematical Model

In order to define a mathematical model of a camera, it is important to define the different coordinate systems. There are three main systems:

World coordinate system The world coordinate system defines the tri-dimensional space on which the whole known universe can be positioned.

Camera coordinate system The camera coordinate system defines a tri-dimensional

space relative to the camera position. A succession of rotations and translations can always be derived to map a point from the world coordinate system to the camera coordinate system. It is the same as the one positioning the camera in the world coordinate system.

Image coordinate system The image coordinate system defines a bi-dimensional space physically implemented by the hardware CCD on the camera where light activates photosensitive cells which are then considered pixels. Mathematically, it defines the plane where points from the World are projected through the camera's mathematical transformation.

Pattern coordinate system The pattern coordinate system defines a bi-dimensional subspace of the world coordinate system that corresponds to the plane where the pattern is located.

2.5 Calibration

The "pinhole camera" model works well in theory, but in real life cameras do not perform perfect perspective projection. Pieces of the camera have to be crafted and imprecisions in the measurements of the light create distortions that have to be compensated through calibration. Calibration models and methods are presented in[22, 36].

Two kinds of parameters have to be found in order to know well enough the camera to actually measure with it. Those are called *intrinsic* and *extrinsic* parameters.

Intrinsic parameters represent the internal workings of the camera that contribute to

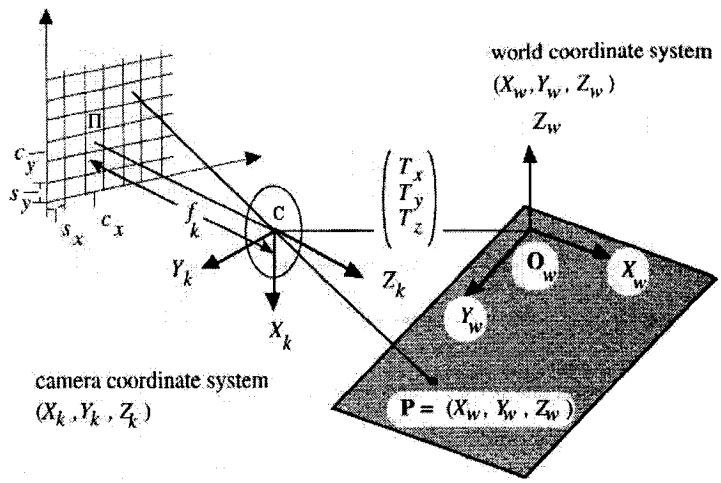

Figure 2.8: Camera coordinate systems [36]

distortion. Here are the main ones:

Focal length Focal length of the perspective projection achieved by the camera.

Principal point The center of the lens (C_x and C_y) and the one of the image are not always aligned perfectly, the real optical center is determined through calibration.

Shear A misalignment of the lens to the retinal plane or a misalignment of the retinal plane themselves will cause a shear in the image when performing perspective transformation. Shear is represented by (S_x and S_y).

Radial distortion The curvature of a lens contribute to add radial distortion to the image as illustrated in figure 2.9.

Pixel size Though pixels are represented as squares on a computer screen, the actual cells

that collect and average the light on their surface are not always. An estimate is usually given in specifications but real sizes in U and V are determined at calibration time.

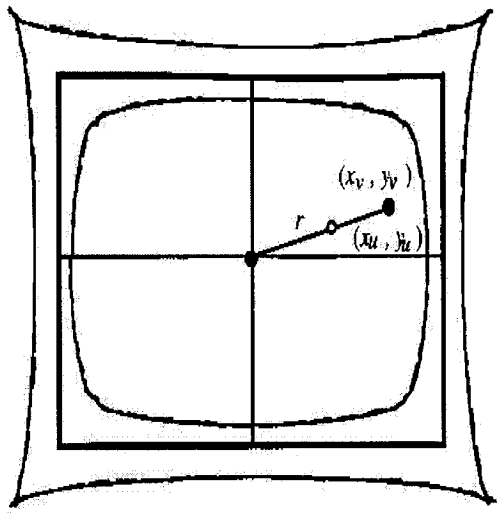

Figure 2.9: Radial distortion [36]

The extrinsic parameters represent the rigid transformation that places the camera in its current position and orientation according to the origin O of the world coordinate system. Basically, it is the rotation and translation that brings the world coordinate systems to the camera coordinate system. The origin is located in relation to the pattern used to calibrate the camera.

2.5.1 Manual calibration

Manual camera calibration is achieved by capturing an image of a pattern with known dimensional properties (Figure 2.10) and then computing the parameters using a set of

linear equations. Since the pattern's pose is known and distance between each target also known, enough equations can be derived to get an over-defined set of equations.

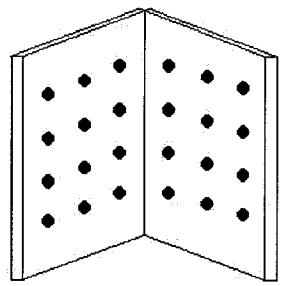

Figure 2.10: Example of a calibration pattern

Based on the number of targets used for calibration and their placement on the pattern, all extrinsic and intrinsic camera parameters can be computed giving, for what we know to date, an accurate model of how the image is captured by the camera.

2.5.2 Auto-calibration

Advances in computer vision introduced a technique, based on projective geometry, to automate this step using features of the tracked pattern as the algorithm is running. This technique, called planar based homographies, allows the mapping of camera coordinates to display screen coordinates [28]. Computation of the homography is presented here followed by the way it is used to provide partial camera calibration in two of the following experiments.

The homography matrix describing the mapping between the camera plane P and the pattern plane P' is computed from four detected targets in image coordinates and four targets in pattern coordinates. A 3×3 homography matrix is a mathematical tool used to

compute the projection from a plane P onto another plane P' when both planes are defined in a 3D space, such that:

$$\begin{pmatrix} x'w \\ y'w \\ w \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.5.1}$$

w being the homogenous scale factor.

Finding the homography that correlates two planes is done by solving a set of linear equations in a similar way to that proposed in [41] but assuming that the last factor of the matrix is equal to 1, therefore reducing the system to 8 equations needed to solve the problem using LU decomposition [48].

$$H = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \tag{2.5.2}$$

Each of the equations is written such that:

$$AC^t = B \tag{2.5.3}$$

where

$$A = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3x_3 & -y'_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 \end{pmatrix}$$

$$C = \begin{pmatrix} a & b & c & d & e & f & g & h \end{pmatrix}$$

$$B = \begin{pmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{pmatrix}$$

This matrix in itself maps any point on one plane onto a point on the other. This is sufficient for some applications like the laser pointing interface (chapter 6), but for other uses like the 6 DOF mouse (chapter 5) individual parameters have to be known.

Auto-calibration through planar homography provides a partial calibration of the camera. Since the homography represents the projection of a plane onto another in 3D space, it can be decomposed into the components that bind those two planes together, namely: translation(Tx, Ty, Tz) and rotation(Rx, Ry, Rz) that bring the two planes in the same referential, the focal lengths along both axis to take vertical and horizontal scaling into account(f_x, f_y) and a scaling parameter λ. How those parameters are extracted is discussed in chapter 5.

It is important to note that the homography does not take any radial distortion into account.

Chapter 3

Tracking through images

3.1 Processing the input signal

Often, the image data coming directly from the camera is not of sufficient quality to do processing. The answer to that problem is to process the image using image processing techniques[15, 45].

Most of the noise comes from fluctuations in the CCD cells. The kind of noise when a pixel's intensity is far from what it should be is called shot noise. It can be dealt with using a median or mean filter. A median filter is generally preferable because it does not dull edges in intensity images, albeit it is more computationally intensive. In binary images though, finding a median is equal to finding a mean.

Thresholding is also used in tracking to create a binary image which states if a pixel is of importance to the tracking algorithm. It is the process by which each pixel is analyzed to see if its intensity is over or under a certain intensity. If it is over, it is set to maximum intensity, else it is set to zero.

Other filters, called morphological operators, are often used in tracking phases. They

include: erosion, dilation opening and closing[45]. Those are used to enhance the quality of an image for tracking.

3.2 Pattern recognition

In order to do any kind of tracking, a known specificity in an image, otherwise known as a pattern, must be found. A pattern is something that defines the area to be tracked by its composition of one or many features. To take an example on the human side, a duck hunter knows the specific characteristics of a duck. When one is in the hunter's field of view, the focus will be on those physiognomical cues to follow its trajectory. In artificial vision alike, it is essential to know the specific features of the pattern, otherwise false detections occur. Those errors are called false positives. Presented next are the most common methods used for 2D pattern recognition, including the one used in this thesis. But first, the taxonomy of a pattern has to be defined.

3.2.1 Rigid vs. deformable pattern

The pattern tracked can be either rigid or deformable. In a rigid pattern, the relative coordinates of one feature to another will always remain constant no matter the context. Take for example a laser dot. It is a pattern formed of one dot that will not change shape over time. An example of a collection of features would be a symbol drawn on a piece of paper. Its features will remain the same over the course of tracking. Opposite to that are deformable patterns. Take for example a hand being tracked without restriction as to its pose. Recognizing such patterns involve more elaborate algorithms that use stochastic methods in order to achieve good performance.

In this thesis, rigid patterns were used because of simplicity of tracking, as compared to the deformable ones. This advantage leads to better processing time, thus increasing update rate and reducing latency.

3.2.2 Natural vs. artificial features

In order to track a point in a scene, one can use either features naturally occurring in the tracked object as a pattern, or introduce artificial ones.

By using the latter approach, since those features are artificial, they can be designed to be easily detectable by contrasting with the surrounding environment. The contrast can be achieved through a difference in light intensity, color, shape or a combination of those factors.

The design of those targets even allows us to detect the centroid with sub-pixel accuracy when their image footprint is larger than 1 pixel. For example, from a target of known regular shape such as a circle, sub-pixel detection can be achieved as follows.

The target is detected and its centroid's position(in pixels) determined. By comparing relative areas over-under and left-right on a pixel basis, it can be determined if the target real center is off. If for example the area on the left of the centroid is larger than the area on the right, and the area over is equal to the area under, the sub-pixel center is centered vertically and on the left in a 3×3 sub pixel grid. Combined with red-blue detection (see section 3.2.6) to double check, this is a robust technique.

A pattern composed of artificial features was chosen for experimentation because of this advantage.

3.2.3 Template matching

There is several ways to recognize a pattern. Perhaps the most intuitive way is to have an internal representation of what the pattern should look like in the image. This way of tracking images is called template matching. A template is basically a rastered image of what the pattern should look like in the camera image [18].

This method is sensitive to noise in the image and to pose of the tracked pattern, if the template is not adapted to different poses, the tracking can fail. But this method can however recognize the pattern at the same time it is detected.

3.2.4 Corner and edge detection

Another way of defining patterns is through corners and edges [41]. A good pattern is composed of features that are rarely found in combination in the surrounding environment where it is to be used. Edges and corners by themselves are everywhere but a combination of them can define a unique pattern.

3.2.5 Blob detection

A simple and quick technique for pattern recognition is to define the pattern as one or many "blobs". A blob is a group of pixels that share a common attribute and a connection. This attribute is often intensity in black and white images or color in color images [45, 55].

When checking for connected pixels, the algorithm can work two ways. It can check for 4-connected pixels or 8-connected pixels.

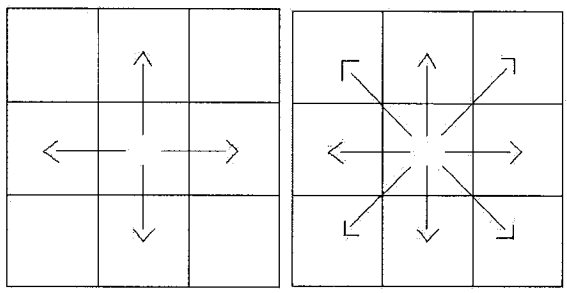

Figure 3.1: Four- and eight-connected blob detection.

3.2.6 Red-Blue detection

A quicker technique developed by the author allows us to detect simultaneously the blob and its centroid. Though it is not as robust as the previous methods, it is well adapted to the task at hand in the following experimentations [39].

The technique exploits the fact that the junction of lines created from the centers of the target on each scan line will define the centroid of a target. To reduce sensitivity to noise, the image is first *binarized* by the use of thresholding. High contrast images such as those in the following experiments provides optimal results. The algorithm is repeated on each horizontal and vertical scan line, marking centers with red for horizontal and blue for vertical:

Now looking for pixels which are red and blue will reveal the centroids. This can be done during the second phase.

Testing was done to compare this algorithm to another one widely used in the field of computer vision. A recursive blob detection algorithm was implemented to compare two factors, time needed to detect the blobs and the percentage of false detections. The two

```
o Switch at off
o For every pixel
    o If switch is off
        o If pixel is white
            o Set switch at on
    o Else
        o If pixel is black
            o Compute center of white stretch and mark it
```

Figure 3.2: Red-Blue detection algorithm.

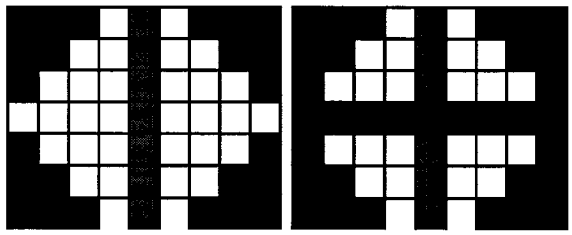

Figure 3.3: Target centroid detection.

algorithms were run sequentially on the same images. Patterns were composed of white circles of 1 *cm* in diameter. Five tests were done with different number of targets: 1 , 16 , 49, 100 and 165 circles. Results on figure 3.4 show that the red-blue detection algorithm gets more attractive as the number of target grows because it takes constant time, compared to linear time for the second algorithm. It also made fewer mistakes in detecting the targets.

Sub-pixel accuracy is reached when detecting the center of the lines. The center is detected at half-pixel accuracy both horizontally and vertically, thus providing quarter pixel accuracy.

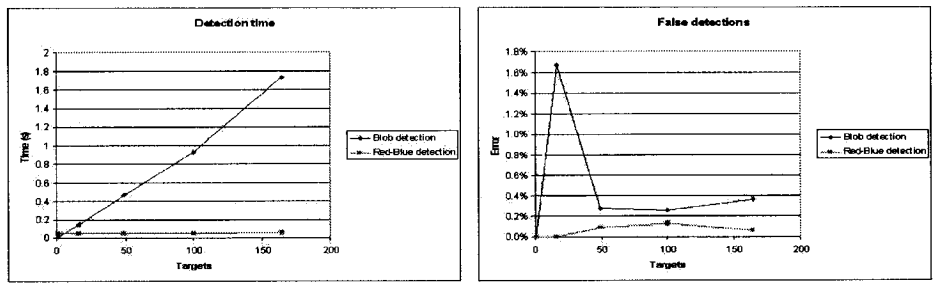

Figure 3.4: Recursive blob detection vs. Red-Blue detection

3.3 Speed optimization

Processing the whole image frame by frame is a computationally intensive task. Most of the time it can be avoided if the delay between every frame is small enough. When an object or pattern is tracked from point A to point B, it does not teleport itself. It does so in a continuous manner. The temporal resolution of motion capture is ultimately limited by the image acquisition rate of the image sensor, i.e. the camera. For a given movement, the faster the camera, the closer the tracked object will be to its previous location in the image. That means that, given a particular speed of motion, by increasing temporal resolution, search for the target can be restrained to a smaller portion of the image, close to the previous position.

Also, if the camera exposition time is too long, motion blur will be introduced in the image which will make tracking with the mentioned method nearly impossible. Motion blur occurs when a moving object's reflected or emitted light is imprinted multiple times on the camera CCDs before the image is integrated.

For practical purposes, the search area is defined as a rectangle, though theoretically it

should be a circle. The size of the rectangle is usually adjusted with trial and error though it obeys the following function.

$$Position(t+1) = Position(t) + Camera\ frame\ delay \times\ speed\ of\ motion$$

An ideal bounding box would be of width and height of $2 \times Camera\ frame\ delay \times speed\ of\ motion$. Assuming the camera is fast enough to avoid motion blur, this would allow optimal tracking. This is an ideal case.

Speed of human interaction can vary greatly, but when a human is accomplishing a task, it is likely that it will move slowly .

The idea of restricting tracking to a portion of the image, when an object is acquired, introduces two phases to the tracking algorithm. An initial phase where the whole image is scanned to acquire the pattern, and a second phase where the target is acquired and is tracked with a smaller portion of the image (figure 3.5).

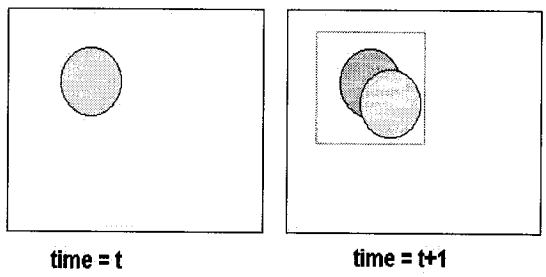

Figure 3.5: Detection range restriction

3.4 Accuracy of the detected pattern position

Since the target is detected with a camera that has a finite number of pixels, tracking accuracy can be limited by that factor. If for example a camera captures images at 640×480 resolution and has a field of view of θ degrees, at z *units* away from the focal point of the camera, every image will cover $h = z \times \tan(\theta)$ *units* horizontally and $v = h \times \frac{480}{640}$ *units* vertically. Similarly, if the image is taken at $2z$ *units* from focal point of the camera, clearly, the area covered will be doubled and precision will be dropped by half.

Because of that precision link with the camera's resolution, a better precision can most of the time be achieved by buying a camera with higher resolution. But higher resolution also means more processing needed for the extra pixels leading to lower *update rate* and higher *lag*. Until processing power is no longer an issue, the idea is to balance the need for precision with the need for speed. In our case, we found that a camera resolution of 640×480 pixels gave good results on our system.

3.5 Experimental setup

All experimentation was done using the following experimental setup:

Table 3.1: Experimental setup

Computer:	Dual Intel P4 Xeon 2.0 GHz , 512 Mb
Frame grabber	Digital frame grabber at 60 fps
Camera	640×480 *pixels* monochrome
High speed camera	1000 Hz monochrome

Measurements were taken with a measurement table describrd in subsection 1.4.2. Accuracy of the table is of ± 0.5 mm in translation and $\pm 0.5^o$ in rotation.

Chapter 4

Reaching into VR: 4-D Interaction

4.1 Device definition

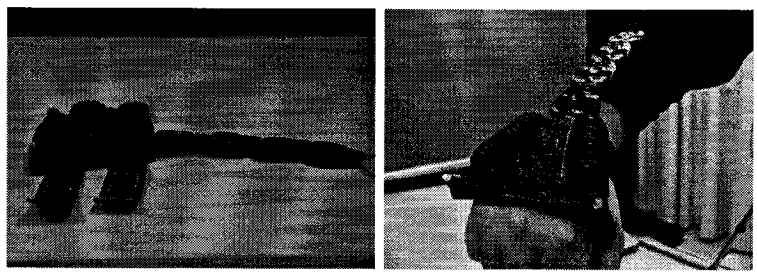

Figure 4.1: The 4-D Mouse

This chapter describes a computer vision-based input device used to provide the position and roll angle of a user's hand. The device was named the four degree-of-freedom mouse (4-D Mouse) and has a simple yet effective design. It is composed of two light emitting diodes (LEDs) mounted on a hand adapted frame and linked to a power source. The working principle behind this device is the recognition of an *a priori* known pattern generated by the two light sources placed at known distance from each other. In the prototype built here, the LEDs are positioned $100mm \pm 0.5mm$ apart (Figure 4.2).

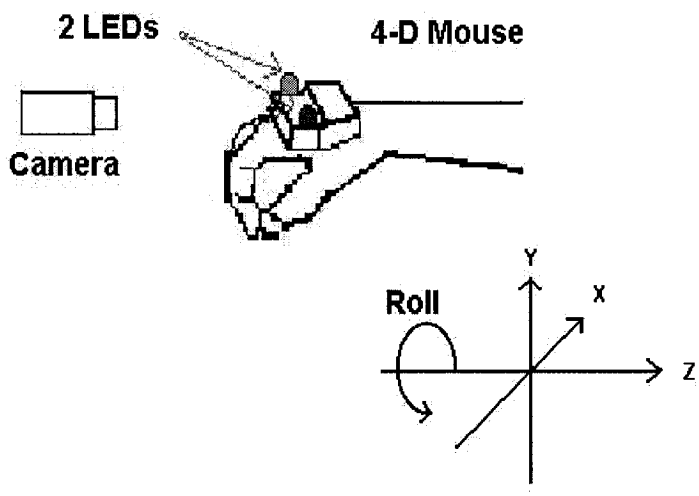

Figure 4.2: 4-D Mouse system

The algorithm used to determine the pose of the device is basically one that tracks a virtual line segment over a series of images, by relying on the two extremities (the LEDs).

Some *a priori* knowledge is needed to make this device work. First, as mentioned before, the distance between the two LEDs on the device must be known. The field of view (FOV) of the camera must also be known in order to compute the depth (Z) component.

The line segment defined by the two LEDs footprint on the camera image is assumed to be on a plane parallel to the retinal plane of the camera, therefore perpendicular to the optical axis. That means that the mouse *must* be perpendicular to the optical axis at all time. In practice this is not always true but, for an open-loop position control task, compensation is made by the brain's sensory-motor system of the user.

The basic idea in processing this input is that the FOV of the imaging sensor forms a square pyramid originating from its focal point. By finding the ratio between the camera's

horizontal resolution (CHR) (in pixels) and the image footprint of the LED baseline (also in pixels), and by multiplying that ratio by the known LEDs baseline in meters, we obtain the total image width (in meters) at that particular $z - plane$.

$$\frac{CHR_{pixels}}{LED\ Baseline_{pixels}} \times LED\ Baseline_m = TIW_m \tag{4.1.1}$$

Then the FOV angle is used to compute the distance between the $z - plane$ and the pyramid tip. (see figure 4.3)

$$\tan(\frac{FOV_x}{2}) = \frac{\frac{TIW_m}{2}}{Z_m} \tag{4.1.2}$$

$$Z_m = \frac{\frac{TIW_m}{2}}{\tan(\frac{FOV_x}{2})} \tag{4.1.3}$$

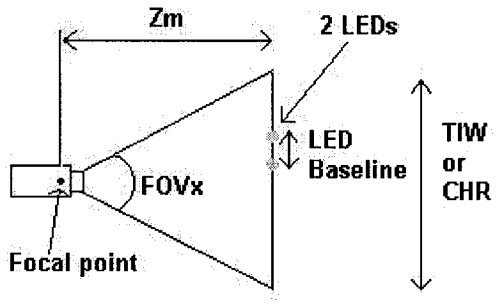

Figure 4.3: 4-D Mouse processing

For instance, if the projected pattern is N pixels wide on the imaging sensor and the horizontal resolution of the camera is CHR pixels, the visible width of the mouse plane is approximately $(CHR/N) \times LED\ Baseline_{pixels}$. And with a horizontal FOV (FOV_x) , the z-plane is at : $Z_m = \frac{(CHR/N) \times LED\ Baseline_{pixels}}{\tan(\frac{FOV_x}{2})}$.

X and Y accuracy depends on the resolution of the camera which, by its discrete nature, limits the position accuracy of the target detection algorithm. Sub-pixel measurements allow us to multiply resolution by a factor and improve measurement. Using an active pattern also introduces other factors impending on accuracy. The camera captures the LEDs as asymmetrical shapes. Light emission creates a halo around the LEDs footprint which creates rounder targets but introduces inaccuracies because of blurring.

The fourth degree of freedom is given by calculating the angle between the projected LED baseline and ahorizontal line parallel to the frame of the image (Figure 4.4). This yields angles with the $xz - plane$ as the plane of origin.

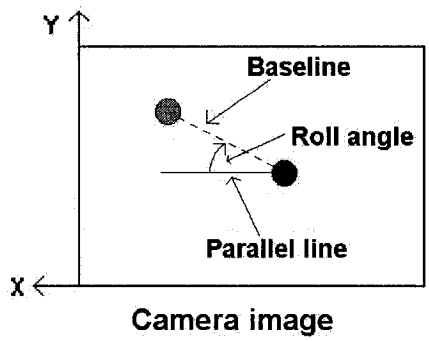

Figure 4.4: 4-D Mouse angle computation

4.2 Targeted applications

Several tasks can be accomplished in the virtual environments using 4 DOF (Position + Roll) devices such as the 4-D Mouse. For example navigation in a virtual environment is done now with off-the-shelf joysticks that have two or three degrees of freedom.

Objects manipulation in VR is a good application for this device. When the user wants to interact with the virtual environment, he can reach for objects that are in front of him, in a single camera setting (as described here), or in whichever direction a camera is, in a multi-camera setting. A multi-camera setting is a setting where position is computed redundantly by more than one camera and where each camera computes an angle. By having the cameras setup in a known configuration, for example three cameras forming an orthogonal setting, this provides extra degrees of freedom on rotation components.

4.3 Experiment definition

Experimentation for this device is divided into three parts, a measure of accuracy, a measure of the update rate and a measure of latency. Resolution of the device is tied to the maximum pixel precision defined in section 3.4.

Position accuracy was determined by using the measurement table (see section 1.4.2). For the X and Y axis, measures were taken at different distances from the camera on a plane parallel to the camera at a depth of 1 m. For Z, measures were taken around 1m over the width and height of the plane. The error was calculated by comparing with real measurements taking $\pm$ the maximum displacement as an estimate of error.

Orientation accuracy was measured by setting and angle manually and moving the device over a plane at 1m and measuring displacement in the measurement and comparing to the real measurement. We took $\pm$ the maximum displacement as an estimate of error.

Update rate was calculated in two ways. First, using a high speed camera set to capture 250 frames/second; by moving the device uniformly and tying the x and y displacement

to a mouse cursor on a projected screen, the difference in time between movements could be measured. Also, the internal computation update rate was measured with an assembly routine based timer. This timer is embedded in the nodes of the *Vision Graph* detailed in appendix A.

To test for *jitter* the device was placed on a tripod in front of the camera and left there, recording displacements in measurements.

Latency was computed by setting the mouse cursor arbitrarily on the screen and placing the unlit mouse in front of the camera, then lighting the LEDs. By capturing the sequence of LED lighting and the screen with the high speed camera at 250 frames/second, the difference in time between the LEDs' lighting to the movement of the mouse cursor on the projected screen could be measured to provide the latency.

4.4 Performance results

Table 4.1: 4-D mouse performance results

Parameter	
Range (Position)	37^o FOV_x, Minimum Z: 13.5 cm, Maximum Z: 1.5m (Camera line of sight)
Range (Orientation)	$\pm 90^o$ with B&W camera, $\pm 180^o$ with color camera
X accuracy	$\pm 1mm$ at $1m$ and 0^o Roll
Y accuracy	$\pm 1mm$ at $1m$ and 0^o Roll
Z accuracy	$\pm 3mm$ when centered. $\pm 5mm$ off-center at 0^o Roll
Roll accuracy	$\pm 2^o$ at $1m$
Update rate	$18Hz \pm 3Hz$ (*Overall*)
	$20Hz \pm 2Hz$ (*Algorithm only*)
Jitter	*Not noticeable*
Latency	$98ms \pm 3ms$

N.B. These results suppose that the 4-D Mouse is perpendicular to the optical axis.

4.5 Analysis

4.5.1 Range

The range is acceptable to work in a VR workbench but better range would be appreciable. Limitation of the range is linked to light intensity and placement of the LEDs on the device. At 1.5m, light emitted by the LEDs is barely distinguishable from the surroundings. To improve range, two things should be done. First, the LEDs that point upwards should be moved to the front of the device, pointing strait in front. Intensity would be perceived more efficiently at greater range by the camera. Second, a light reflecting material should be placed at the back of the LEDs to redirect light emitted towards the back of the mouse to the front. This should improve intensity, thus improve the range.

4.5.2 Accuracy

Accuracy of the 4-D Mouse is better at $1m$ than the accuracy of the 6-D Mouse (see chapter 5) and informal testing shows that it is also true at a higher distances. But accuracy depends on the accurate calculation of the FOV parameter.

4.5.3 Update rate

At $20Hz$, the update rate is better than the 6-D Mouse. This is due to the simplicity of the algorithm. On the other hand, to use this device, a parameter of the camera must be known. It can be measured by taking measurements of image width at several distances from the camera. The difference in width over the difference in distance from the camera is equal to the sine of the FOV angle.

4.5.4 Jitter

Jitter is not noticeable at the test distance. Informal testing shows that jitter is not noticeable between minimum and maximum range. Therefore, filtering the output of this device before using it as a tracker is not necessary.

4.5.5 Latency

Latency is just under 100 ms which should hamper user's performance only slightly, because it is a little over the threshold for acceptable usability of 80 ms reported in literature [62].

4.5.6 Infrared vs. Visible Pattern

In terms of numerical results, it does not matter wether the LEDs emit in the visible or infrared wavelength range. The difference shows in the sensitivity to the environment.

Using visible light LEDs, the lights have to be turned off for the device to work properly. Using IR LEDs coupled with an interferential filter on the camera, light radiation from a normal lit interior working environment is cut out to leave only the infrared pattern. In this condition the device works as well as with the lights turned off/visible light LEDs setup. The downside is that, even with a color camera, there would be no easy way of discerning the left LED from the right one except maybe by putting a light absorbing filter on one of the LEDs to reduce the brilliance enough to differentiate. Also, the camera used must be sensitive to near-infrared wavelengths. Commercial off-the-shelf webcams often use CMOS cells to capture light and create the image. CMOS cameras are very sensitive to light in the red spectrum. For that reason, a filter is generally added to attenuate red in the image. This often cuts out most of the infrared as well, therefore making them unsuitable for the use with IR LEDs.

Using IR LEDs makes the device almost invisible to the human eye in a dark environment, thus enhancing the feeling of immersion or *Presence* in virtual environments.

4.6 Discussion & future work

This chapter described the design and implementation of a 4 degrees of freedom hand tracking interface for use in virtual environments. This design provides a light and low cost alternative to traditional position trackers when all six degrees of freedom are not required to perform a task.

This device has been designed to provide X, Y, Z, *Roll* coordinates but moving the camera to another position, for example overhead, can change the *Roll* data into *Heading*

data which might be more relevant for a specific application. This is also true with the camera on the side to obtain *Pitch* data. It could also be useful to use three instances of the device and three cameras to compute position redundantly, therefore more robustly, and the three orientation components.

The main downside is that the user must always keep the pattern perfectly perpendicular to the optical axis of the camera, which is sometimes difficult to achieve.

Even with batteries, the device is very light, it is easy to forget that it is there when working with it. This transparency is also an asset in VR applications.

Chapter 5

Full pose control: 6-D Mouse

5.1 Device definition

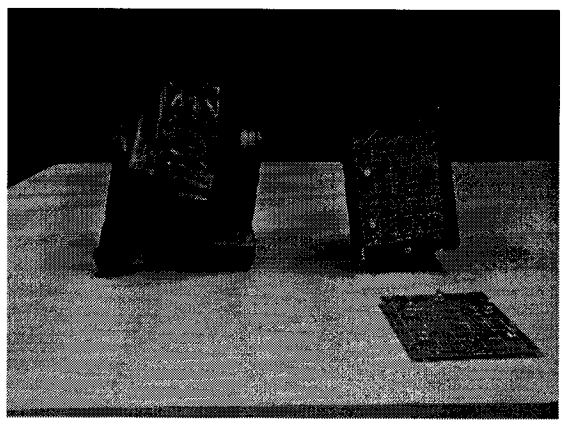

Figure 5.1: The 6-D mouse prototypes: 1(left) and 2(right)

The six degree-of-freedom mouse was first presented in 2000 by Kumar [37]. In 2002, an enhanced version of this mouse (Prototype 1 in figure 5.1) was presented by the author of this thesis [39]. The second prototype (Prototype 2 in figure 5.1) was then built with a larger interchangeable pattern and different light sources (LEDs instead of micro-lamps)

. It is basically an active pattern of LEDs detected by the Red-Blue target detection algorithm presented earlier (see section 3.2.6) and processed using planar homographies to detect position and orientation of the device with six degrees of freedom. The homography technique was chosen over stereo methods because it requires only one camera input. Using two cameras doubles the processing time thus reducing update rate and introducing more lag. It also doubles radial distortion factors that are not taken into account in auto-calibrated methods. Stereo, on the other hand, provide more accurate depth measurements and no undefined cases when the pattern is parallel to the camera.

The LED pattern originally used by Kumar remained the same (figure 5.2). Five LEDs (L_1 to L_5) are used to compute position and one LED (L_t) is used to detect a trigger.

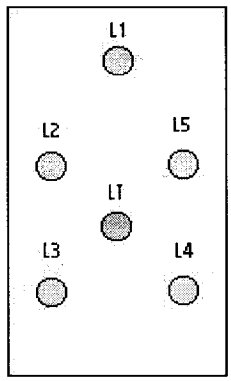

Figure 5.2: the 6-D Mouse pattern

The mouse itself was modified in the following way to make it more suitable for the VR Workbench:

- The light sources were moved from the top of the hand to the side to limit the visual interference in the user's FOV, to preserve the feeling of immersion.

- The mouse was covered with black material to make it harder for human eyes to detect it in a dark environment. It is then convenient for the user and other persons in the room that will not see the device as well.

5.1.1 Pose estimation

To calibrate the camera (see section 2.5), a target pattern is usually used to find the extrinsic and intrinsic parameters of a camera. The extrinsic parameters, which define essentially a translation and a rotation, are calculated with the calibration pattern as an anchor point into the world coordinates. That means that position and orientation of the camera are computed relative to the calibration pattern. Now, let the camera be static and the pattern move around. Calibration techniques would then give at a given point in time, position and orientation of the camera relative to the pattern, or in another perspective, position of the mobile pattern relative to the camera. This technique is used for registration[1] [41, 42] and for tracking input devices [37, 39].

The homography based auto-calibrated method, as described in section 2.5.2, was chosen to compute pose. For this algorithm, four out of the five targets on the mouse are used to compute the homography, $L2$ to $L5$. $L1$ is only used to determine which ones are which in the detection phase. It is also assumed that the center of the image is at the optical center, or principal point, of the camera as in [37].

Position estimation From the matrix H found at each iteration (see section 2.5.2) can be derived all the information needed to know the exact pose of the object. The position is

[1]Registration is the process of finding the optimal translation and rotation parameters to overlay a synthetic image onto the real world so that they appear to ba parts of the same one.

estimated using parameters h_{13}, h_{23} and h_{33} along with two hybrid intrinsic parameter (f_u and f_v) as well as the scaling factor λ. There is only one focal length in the camera but f_u and f_v also take some scaling factors into account. According to [41], those three factors are calculated as follows:

$$f_u = \sqrt{\frac{h_{11}h_{12}(h_{21}^2 - h_{22}^2) - h_{21}h_{22}(h_{11}^2 - h_{12}^2)}{-h_{31}h_{32}(h_{21}^2 - h_{22}^2) + h_{21}h_{22}(h_{31}^2 - h_{32}^2)}} \tag{5.1.1}$$

$$f_v = \sqrt{\frac{h_{11}h_{12}(h_{21}^2 - h_{22}^2) - h_{21}h_{22}(h_{11}^2 - h_{12}^2)}{-h_{31}h_{32}(h_{11}^2 - h_{12}^2) + h_{11}h_{12}(h_{31}^2 - h_{32}^2)}} \tag{5.1.2}$$

$$\lambda = \frac{1}{\sqrt{h_{11}^2/f_u^2 + h_{21}^2/f_u^2 + h_{31}^2}} \tag{5.1.3}$$

The translations vector $T = (t_x, t_y, t_z)$ can then be computed directly:

$$t_x = \frac{\lambda h_{13}}{f_u} \tag{5.1.4}$$

$$t_y = \frac{\lambda h_{23}}{f_v} \tag{5.1.5}$$

$$t_z = \lambda h_{33} \tag{5.1.6}$$

Orientation estimation There are two ways to determine the orientation. The one proposed by Kumar [37] uses the different angles between the LEDs to compute the orientation. The second proposed here and in [39] uses the already available homography matrix to compute orientation.

For the matter of this experiment, the latter was chosen because it is intrinsically faster. The rotation given by the homography is a composite of the rotations along the three axes. Dividing it along the three main axes is done as follows:

Let $\theta_1, \theta_2, \theta_3$ be respectively the rotation angles around the X_c, Y_c and Z_c axes. These rotations are defined with the following 3×3 matrices [25]:

$$R_X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 \\ 0 & \sin\theta_1 & \cos\theta_1 \end{pmatrix}$$

$$R_Y = \begin{pmatrix} \cos\theta_2 & 0 & -\sin\theta_2 \\ 0 & 1 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 \end{pmatrix}$$

$$R_Z = \begin{pmatrix} \cos\theta_3 & -\sin\theta_3 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

In order to simplify writing, let:

$$\begin{aligned} s_1 &= \sin\theta_1 \quad c_1 = \cos\theta_1 \\ s_2 &= \sin\theta_2 \quad c_2 = \cos\theta_2 \\ s_3 &= \sin\theta_3 \quad c_3 = \cos\theta_3 \end{aligned}$$

Rotation order is irrelevant because the center of rotation remains the same. Therefore R can be formed as follows:

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_1 & -s_1 \\ 0 & s_1 & c_1 \end{pmatrix} \cdot \begin{pmatrix} c_2 & 0 & -s_2 \\ 0 & 1 & 0 \\ s_2 & 0 & c_2 \end{pmatrix} \cdot \begin{pmatrix} c_3 & -s_3 & 0 \\ s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.1.7}$$

Solving equation 5.1.7:

$$\begin{pmatrix} c_2c_3 & -c_2s_3 & -s_2 \\ -s_1s_2c_3 + c_1s_3 & s_1s_2s_3 + c_1c_3 & -s_1c_2 \\ c_1s_2c_3 + s_1s_3 & -c_1s_2s_3 + s_1c_3 & c_1c_2 \end{pmatrix} \tag{5.1.8}$$

From there

$$r_{13} = -s_2 = -\sin\theta_2 \tag{5.1.9}$$

thus

$$\theta_2 = \arcsin(-r_{13}) \tag{5.1.10}$$

From there θ_1 is derived from r_{23} and θ_3 from r_{12} :

$$\theta_1 = \arcsin\left(\frac{-r_{23}}{c_2}\right) \tag{5.1.11}$$

$$\theta_3 = \arcsin\left(\frac{-r_{12}}{c_2}\right) \tag{5.1.12}$$

Next, from [41] we know that:

$$R = \begin{pmatrix} \lambda h_{11}/f_u & \lambda h_{12}/f_u & r_{21}r_{32} - r_{31}r_{22} \\ \lambda h_{21}/f_v & \lambda h_{22}/f_v & r_{31}r_{12} - r_{11}r_{32} \\ \lambda h_{31} & \lambda h_{32} & r_{11}r_{22} - r_{21}r_{12} \end{pmatrix} \tag{5.1.13}$$

then equations that define rotation are derived from the homography matrix. First using equation 5.1.13 in equation 5.1.11:

$$\theta_1 = \arcsin\left(-\frac{\lambda^2}{f_u c_2}(h_{31}h_{12} - h_{11}h_{32})\right) \tag{5.1.14}$$

then using equation 5.1.13 in equation 5.1.10:

$$\theta_2 = \arcsin\left(-\frac{\lambda^2}{f_v}(h_{21}h_{32} - h_{31}h_{22})\right) \tag{5.1.15}$$

finally using equation 5.1.13 in equation 5.1.12:

$$\theta_3 = \arcsin\left(-\frac{\lambda^2 h_{12}}{f_u c_2}\right) \tag{5.1.16}$$

5.2 Targeted applications

The targeted applications for the 6-D mouse are many. It is a tool that can be used for mostly any one handed task in three dimensions, like what the standard computer mouse is to the two dimensional screen.

It can be used for navigation providing all six degrees of freedom, if not all degrees are used, some can just be ignored to avoid coupling.

It can be used for selection and manipulation of an object in three dimensions with the restriction that a line of sight is needed between the camera and the mouse's LED pattern.

5.3 Experiment definition

Experiments on the 6 DOF mouse were exactly the same as on the 4 DOF mouse except that they were conducted at about 30 cm from the camera and that pitch (R_x) and yaw (R_y) rotations were also tested.

5.4 Performance results

For the following tests, the second prototype (with LEDs), was used.

Method Since the focal length of the camera is auto-calibrated at the first frame and fixed for the rest of the execution, some steps have to be taken in order to get better results:

- If the pattern plane is aligned with the camera image plane, the focal length is not defined, thus, having the mouse tilted at an angle for the first image frame is necessary.
- Assuming that the optical center of the camera is aligned with the center of the image,

the effect of the radial distortion should be more uniform when the pattern is centered. Holding the mouse centered in the image for the first frame yields better focal length computation.

Table 5.1: 6-D mouse performance results

Parameter	
Range (Position)	37^o FOV_x, Minimum Z: 14 cm, Maximum Z: 1.0 m (line of sight)
Range (Orientation)	Pitch: $\pm 180^o$, Roll/Heading: $\pm 45^o$
X accuracy	$\pm 6mm$ at $30cm$ and RX, RY, RZ at 20^o
Y accuracy	$\pm 3mm$ at $30cm$ and RX, RY, RZ at 20^o
Z accuracy	$\pm 8mm$ when centered at Z=30 cm $\pm 5cm$ with RX, RY, RZ at 20^o.
RX accuracy	$\pm 11^o$ at $30cm$
RY accuracy	$\pm 11^o$ at $30cm$
RZ accuracy	$\pm 3^o$ at $30cm$
Update rate	$11Hz \pm 7Hz$ (*Overall*)
	$14Hz \pm 1Hz$ (*Algorithm only*)
Jitter	$1Hz$ to $3Hz$ causing $0.2mm$ and 3^o uncertainty at 30cm
Latency	$150ms \pm 50ms$

5.5 Analysis

5.5.1 Range

Even though range of this tracking device is better than the 4-D mouse, at a range of about 1.0 m, noise in the measurement becomes too high to use this data reliably without filtering.

5.5.2 Accuracy

The accuracy of the device is debatable. If the focal lengths (f_u, f_v) are not acquired accurately at the beginning, pose accuracy will not be totally accurate. Inaccuracies come from the homography calculation, which is sensible to noise in centroid detection. Using a bigger pattern would provide better centroid estimation, therefore a more accurate homography

calculation. The device then could not be used close to the camera therefore brighter and/or bigger targets would have to be used.

Another major factor acting against accuracy is radial distortion. In Kumar's paper it is said to be negligible. In practice, plotting real measurement vs. provided measurements shows a curve that is interpreted as a symptom of radial distortion of the lens (Figure 5.3). This means also that if the footprint of the pattern in the camera image is large, target positions are displaced giving erroneous measurements. A method or procedure for auto-calibrating for radial distortion would improve accuracy of this tracker.

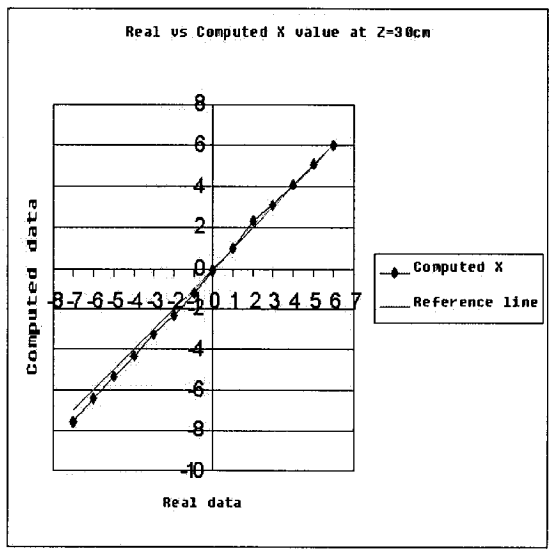

Figure 5.3: Error along X axis for the 6DOF mouse

Although the R_x accuracy reported in a previous paper is bad with $\pm 23^o$ at 30 cm [39], we found that it was due to the fact that the focal length was measured with the pattern's footprint not centered in the image. If the focal length is computed with the pattern's

footprint close to the center of the image, then the R_x accuracy improves to $\pm 11^o$ at 30 cm as indicated in table 5.1.

Also, R_x accuracy improved when the 6-D Mouse was farther from the camera. It is postulated that, because the footprint of the mouse pattern is smaller on the image, radial distortion does not have as much an effect over measurement at large distance.

5.5.3 Update rate

With $11Hz \pm 7Hz$, the update rate is nearly at a point where interaction is comfortable. It is limited by computation speed and thus a faster computing platform will solve the problem. Little improvement can be expected from the algorithm because the most computationally expensive part is the solving of the linear equation system which is theoretically done optimally in [48].

5.5.4 Jitter

Jitter is important with the device. First, if the focal length is not constant through the execution, pose data is not usable because of jitter noise. For this experiment, focal length was fixed at the first frame. Second, for the focal length to be as accurate as possible, the pattern has to be held in a position that is not parallel to the plane of the camera. Techniques to avoid this jitter is used in [41] but it involves using a RANSAC algorithm with more points on a pattern to stabilize the system [23]. This cannot apply to this device because the number of points on the pattern is limited.

5.5.5 Latency

With $150ms \pm 50ms$, the latency obtained is greater than for the 4-D Mouse described in the previous chapter ($98\ ms \pm 3\ ms$). This can be explained by the fact that the algorithm is more computationally expensive than in the other case. This might also point out that there is a bottleneck in the event management in the Vision Graph framework (See appendix A).

5.5.6 Infrared vs. Visible Light LEDs

Like for the 4-D Mouse, the use of infrared LEDs did not improve measurements but has the benefit of making the system more robust to the environment with a higher signal to noise ratio (SNR). The lower SNR provided by the visible LEDs caused the tracker to be unusable under normal room lighting while it was possible with the IR LEDs and an interferential filter.

5.5.7 LEDs vs Micro lamps

It seems from the two prototypes that micro lamps provide a smoother targets than LEDs. LEDs, especially the IR ones produce specular-like reflections on the camera lens. Even nearly closing the iris of the camera does not solve the problem. That problem was solved by putting a light diffuser over the LEDs made of a piece of white nylon. This provided smoother targets to track and increased target detection rate.

5.6 Discussion & future work

This chapter presented an algorithm that can be used to track position of an LED pattern with 6 degrees of freedom. The concept was implemented in a handheld device with the intent of using it as a hand tracking device in a virtual reality room environment.

Performance in terms of pattern detection is good as long as there is no light source brighter than the LEDs. This condition is not met when sunlight shines in front of the camera or when bright indoor light sources appear on the image. Using an infrared LED pattern generally fixes the problem for indoor lighting allowing the mouse to be used in a lit room. This introduces the same restriction as for the use of CMOS webcams as seen in section 4.5.6.

Using a color camera is not necessary but will help in the process of identifying LEDs before the auto-calibration process if different LED colors are chosen for the pattern.

Comparing this device to competing technology reveals advantages and disadvantages. The main advantages this technology has are:

Easy setup: This technology does not require any special setup. Once the pattern is visible on the camera image and the focal length determined (see section 5.4), position data will be acquired.

Low cost: Not counting the price of the camera, this 6 DOF device's cost is negligible compared to all the other technologies.

The main disadvantages are:

Variable accuracy: Depending on the camera system used to capture the image, accuracy may vary. That is mainly due to radial distortion induced by low quality lenses. Radial distortion is not accounted for in planar homography based auto-calibration.

Necessity of a line of sight: This devices requires a line of sight with the camera. This restricts user movements but could be improved by using several cameras to cover more angles.

Filtering data could reduce noise and increase the usable range to 2 m from the camera. At ranges greater than 2 m, target detection rate is too low to provide reliable measurements.

More research is needed to assess if some LED patterns are better than others and to see if using more LEDs could reduce the effect of radial distortions as well as jitter.

Chapter 6

2-D in 3-D: Laser Pointer

6.1 Introduction

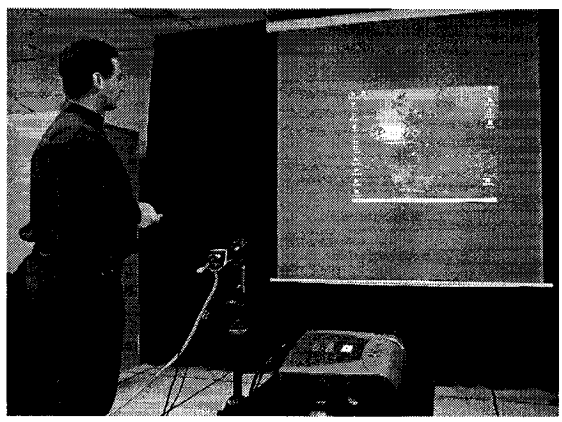

Figure 6.1: The laser pointer interface

One of the most used device during public presentations is the laser pointer, mainly for pointing and highlighting specific parts of the displayed image on a projection screen, particularly when it is out of reach of the user's arm. Unfortunately, used in this way, laser pointers[1] are just passive devices that offer no possibility of interaction with the content

[1]To avoid confusion in this chapter, the term *pointer* describes the laser pointer and the term *cursor*, the "mouse" pointer on the screen.

displayed on the screen.

In the recent years, several publications have been made describing computer vision based systems that use a laser pointer as an input device [19, 20, 21, 35, 43, 44, 32, 47, 63].

All the techniques described so far to track a laser spot on a screen require the previous calibration of the camera. We present here a technique based on planar homographies geometry that eliminates this step and thus simplifies the setup process.

We then present a detailed performance analysis of an implementation of the system in order to check if there is a match with the known basic requirements for effective human-computer interaction.

The analysis is performed under several conditions, including the use of both a visible and of an invisible (infrared) laser pointer.

6.1.1 Previous work

One of the first papers on laser pointers is from Kirstein and Müller and describes the use of laser pointer as an input device and gives some advice on its implementation in [35].

The idea was taken further by Olsen and Nielsen who compared user performance between the mouse, a physical device and a laser pointer for a complex task involving pointing at widgets and manipulating their functions [32]. The results indicate that the mouse was twice as fast as the other two input devices but the control variables such as the update rate (and thus latency) of the different input devices were significantly different. The laser pointer was disadvantaged by its low update rate (7 Hz compared to an estimated 40 Hz for a standard mouse), as well as the typical training bias induced by subjects experience

with the mouse. In fact, in another study where the latency was comparable between both devices, the laser pointer was found to perform as well as the standard mouse [19]. It is important to note, though, that this latter user study was conducted with only four subjects.

Target acquisition, i.e. pointing and selecting, by using a laser pointer with an extended *laser-on* (similar to a *mouse-over*) technique has been tested and proven inaccurate. This was mainly due to jitter in the laser spot caused by the user's hand wiggle [43, 47]. To reduce jitter in the measured position, an averaging filter was used to estimate pointing position over $\frac{1}{2}$ second. This approach slightly reduced the jitter but at the expense of adding $\frac{1}{2}$ second to selection time, in addition to the extended laser-on period.

A potentially better way to use laser pointers is to use semantic snarfing [43, 44]. This technique allows the user to point at an area of interest which is then transferred to a secondary device such as a *Portable Data Assistant*(PDA) so that the user can point more precisely. According to these papers, semantic snarfing is faster but prone to errors because the interface is scaled down due to the PDA's small screen size. A proposed solution to avoid the latency introduced by the extended *laser-on* technique is to add a push-button on the laser pointer or for use by the other hand.

A laser pointer equipped with a push-button can also be used as a pen when the screen is translucent and the camera located behind it [63].

Concerning the laser pointer, one study compared the performance of a visible and invisible laser and found that the user response time was better with the visible one [19]. This result was explained by the fact that the user focused on the laser spot rather than on

the cursor, thus eliminating computer display latency and providing an ideal feedback loop between the motor and perceptual systems.

In this chapter, a method to implement an auto-calibrated laser pointer system using planar homographies is presented. A similar idea was also put forward but not detailed in [58] for the purpose of 2D presentations.

Finally, we measure the performance of a system implemented with this technique and compare the perceived system responsiveness when using visible (red) vs. invisible (infrared) laser pointers. The proposed thesis is that a visible pointer will make system latencies more perceptible because of the visible gap between the laser spot and the cursor, especially during fast movement.

6.2 Laser pointing algorithm

6.2.1 Assumptions

The system works as long as the three following conditions are filled:

- The screen is entirely visible (and in focus) inside the field of view of the camera
- There is no relative movement between the screen and the camera once the system is started
- The laser spot is brighter than the rest of the screen

6.2.2 System setup

To test the algorithm, a camera with an image sensor of 640×480 pixels and which can capture images at 60 Hz is used for input. For display on the projection screen, an XGA

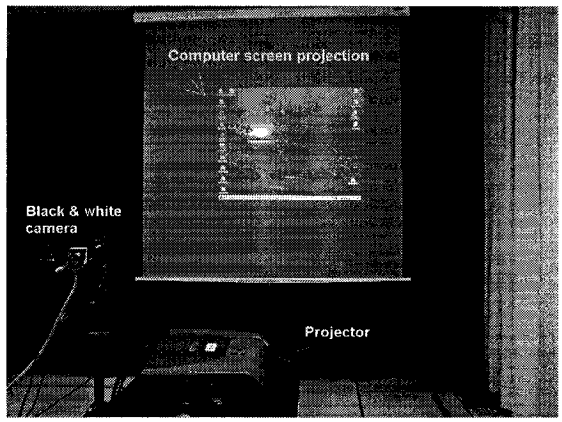

Figure 6.2: The laser pointing system.

(1024 × 768) DLP projector with a refresh rate of 85 Hz is used. Two laser pointers are tested. A visible (red) laser pointer for the user to see the actual laser spot, and an invisible (infrared) laser pointer that creates a laser spot outside of the human's perceptible light spectrum.

The camera is located in a way that the whole projected display fits in the camera image. All camera auto-adjustments are turned off. Also, for one of the experiments, with the infrared laser, an infrared interferential filter is mounted on the camera after the preprocessing phase, in order to block any light outside of the infrared range, and thus ease the laser spot detection process.

6.2.3 Preprocessing

The first step is to estimate the pose of the screen in order to accurately compute the pointer position. The position of the four corners of the computer screen is detected and stored since it defines the bounding box for future frame processing. To compensate for the

phase difference between the projector and the camera, the binary image of the screen is accumulated over ten frames to get a more accurate estimate of the corners.

The homography matrix describing the mapping between the camera plane P and the display screen plane P' is then computed from the four detected corners in image coordinates and the screen four image screen corners in screen coordinates. A 3×3 homography matrix is, as described in section 2.5.2, a mathematical tool used to compute the projection from a plane P onto another plane P' when both planes are defined in a 3D space.

The edges of the screen's footprint in the camera image are computed as 2D vectors to test wether or not the laser spot is inside the projected screen.

6.2.4 Processing algorithm

First, form a binary image by using a threshold to segment the image.

Second, find the laser spot in the image by computing the centroid of the bounding box of the brightest spot on the screen. The laser spot is supposed to be the only part of the image with a high light intensity. Also, since the laser spot cannot be assumed to be of a specific shape because of light leaks, the use of a bounding box is necessary.

To eliminate laser spots located outside the displayed screen, the centroid of the laser spot is compared to each of the four edges of the screen to determine if it is inside. If it is the case, the homography is applied to the detected laser point to bring it into the pattern space, and therefore in screen coordinate space.

We finally transform the screen coordinates space into a cursor position (in pixels).

6.3 Targeted applications

As a two DOF input device, laser pointing can be applied in virtual environments for object selection and manipulation. It can be defined as a *initial line of sight* interface since that, in order to consciously interact with something in the virtual world using this interface, the user has to see it to point at it. Subsequent actions taken by the user will not necessarily involve keeping line of sight, just a continuity in the user's task. In HCI terms, it is well suited for a selection task and a manipulation task.

For example, let us say that a task is to take a virtual sheet of paper and insert it in an envelope. The user initially takes the sheet of paper by pointing at it and then, dragging it on the same plane as the envelope, inserts it in the envelope. At the specific point where the sheet of paper is completely inside the envelope, line of sight is broken but not continuity of the task or immersion in the virtual environment because it *makes sense.* This tasks involves a two-dimensional operation in a three dimensional world.

6.4 Experiment definition

The implemented system was evaluated under three ambient lighting conditions: dark room, normal indoor room and sunlit room.

Also, the system was tested with two laser pointer wavelengths: 680nm (red) and 820nm (infrared). And in the IR laser case, we tested 2 camera settings: one with an interferential filter and the other with no filter.

The projector and the camera were located at approximately at 2 m of the screen and the optical axes were 15^o apart. The user was behind the setup at approximately 2.5 m

in front of the screen. At that pointing distance, the visible laser spot from a commercial laser pointer is 2 *cm* in diameter at its largest.

The metrics used to evaluate the performance of the system are: range, accuracy, update rate, jitter and latency.

The *accuracy* was calculated by pointing at different regions of the screen and recording the difference in pixel from the tip of the cursor to the laser spot. This was done only with the visible laser pointer.

The *update rate* was calculated by moving constantly the laser spot on the screen and by measuring the time difference between cursor updates in a sequence of images. That video sequence of the scene was recorded by a high speed (250 *fps*) video camera.

To test for *jitter*, the pointer was placed on a stable surface and pointed at the screen while the displacements were recorded by a video camera.

To measure *latency*, the cursor was placed at an arbitrary position in the computer screen. The pointer was then activated to point at an arbitrary region within the screen area. The time taken by the cursor to reach the static laser spot was then measured from a recorded video of the event by counting the number of frames between the two events. The video of the scene was again recorded by the high speed camera.

6.5 Performance results

The system performed well under under the dark room and normal indoor conditions. Under direct sunlight however, the system could not detect the screen because of a lack of contrast between the projected image and the rest of the screen.

Under the other two lighting conditions (normal indoor and dark room),the performance results are summarized in Table 6.1 :

Table 6.1: Laser pointer performance results

Parameter	Result
Range	*Line of sight*
X accuracy	± 2 *screen pixels average*
Y accuracy	± 3 *screen pixels average*
Latency	$52\ ms \pm 8\ ms$
Update rate	$60\ Hz \pm 1\ Hz$ (*Overall*)
	$605\ Hz \pm 5\ Hz$ (*Algorithm only*)
Jitter	$1\ Hz\ \pm 1$ *screen pixel*

It is interesting to note that from informal testing that, even when the camera is off by up to 30^o from perpendicular axis of the screen, the system works properly.

6.6 Analysis

6.6.1 Range

Range of the system is limited by: the need for a *line of sight* between the pointer and the screen, between the camera and the screen and the need for good contrast between the laser spot and the display. It is important to note here that at high distances, a laser pointer will project a weaker laser spot lowering the contrast to a point where it might not be usable.

In practice however, range is basically limited by factors such as user's pointing performance, and depends on its performance. As long as the projector projects an image that is bright enough to contrast with the environment and as long as the footprint of that image in the camera is good enough to get the desired resolution and that the user can point accurately, the system will work. For most uses, a range of up to 10 m is achievable and should be sufficient since it is roughly the size of a presentation room.

6.6.2 Accuracy

Accuracy of the laser pointer device is relative to the size (in pixels) of the laser spot on the screen and captured by the camera. If the camera is closer, the laser will be integrated over several pixels on the camera's retinal plane, thus providing a better estimate of the laser spot's centroid.

Also, laser diodes have *spreading* factors. Laser diodes used in commercial laser pointers tend to leak more on the sides of the spot than higher quality diodes. This influences centroid computation.

Thirdly, the angle between the projected screen and the camera will have influence on the accuracy and uniformity of the results. If the angle is great, one pixel in the camera will map to several pixels on the screen. When the angle between the camera plane and the display screen plane reaches 30^o, inaccuracies are is to a point where precision tasks are impossible on the part of the screen farthest from the camera.

Fourthly, resolution of the system has an influence on accuracy performance as pixels on the projected screen have to be differentiable if one is to point at them.

Resolution is limited by the camera's resolution, the *screen to camera* distance and the screen size. Resolution causes inaccuracies as long as the *effective image resolution*, i.e. resolution taking into account the sub-pixel factor, is below the projected image's resolution. On the other hand, if the resolution is too high, movement detection do not necessarily result in cursor movement and useless *cursor move* events are generated.

Detecting the centroid of the bounding box of the laser pointer enables us to achieve

up to quarter-pixel resolution, depending on the laser spot size in pixels, increasing the effective camera resolution four-fold over native resolution.

To achieve optimal resolution, the camera should be placed so that the retinal plane is parallel to the projected image plane, and that the four boundaries of the auto-detected projected image are parallel to the corresponding camera image boundaries. Then, since the detection algorithm permits twice the resolution along both axis, the horizontal and vertical camera resolutions should be half the size in pixels of the detected projected image.

Under the experimental conditions described above, the 640×480 camera provided enough resolution to detect one *screen pixel* movements.

Finally, from testing the system, it seems that radial distortion in the camera lens has a noticeable effect on the measurement. Accuracy is better in the center of the camera image than on the sides. Also, inaccuracies tend to shift towards the screen sides. For example, the cursor tends to be a little higher than the laser spot when in the upper portion of the screen and a little under when in the lower portion of it.

In summary, the accuracy depends of five factors:

- Distance between the camera and the screen
- Light leaking of the laser pointer
- Angle between the camera and the projector
- Effective camera resolution
- Radial distortion of the camera lens

6.6.3 Update rate

At 60 Hz the update rate is good enough for interaction, and induces a latency that does not hinder user performance. Clearly here, the update rate is limited by the camera speed (in frames per second).

6.6.4 Jitter

At ± 1 screen pixel, jitter from the device is limited as compared to natural human hand wiggle as shown in informal trials. The wiggle is estimated to be ± 3 screen pixels for the particular setup used here and with an average user located at a distance of 2.5 m .

6.6.5 Latency

Latency is less than twice the update. However, with an overall system latency of 52 $ms \pm$ $8ms$ is well under the limit of 80 ms which has been shown to hamper user performance in tracking tasks with position control [62].

6.6.6 Infrared vs. visible laser pointer

Tests were made using a visible and an invisible laser pointer. Although the system performance is the same in both cases, the camera used seems to be more sensible to infrared providing a better contrast with the projected screen, even without the use of an interferential filter. For the user, there is a big difference in terms of adaptation. When using the red laser pointer. The user sees the red laser spot on the screen and expects the cursor to follow directly, thus yielding a high sensitivity to latency, update rate and accuracy. When using the IR laser pointer, the laser spot is not visible and the human sensory motor system

adapts to the latency and correct for the inaccuracies through open loop adaptation. The system seems more fluid though it reacts at the same speed. It is important to note though that one previous study [19] indicates that the IR pointer reduced the user performance for pointing tasks due to the longer response time. It is therefore possible that a visible pointer could perform better albeit giving the impression of greater latency. But a user study is necessary to demonstrate that and is not within the scope of this work.

6.7 Discussion & future work

This chapter presented and evaluated a new computer vision based algorithm that can be used to track a laser spot on a projected screen, in order to use laser pointers as an input device. One application possible of this system is for group presentations on a large projected screen, but virtual reality and augmented reality systems could also benefit from such an interface.

We found that performance of the pointing system is sufficient to be used in most applications. Resolution could become a hardware problem if the ratio *display resolution* over *effective camera resolution* is not kept close to 1. This leads to an oversensitivity of the system if the displayed screen's footprint in the camera image is in higher resolution than the actual displayed image. The system then reacts to movements in the laser pointer that would not lead to an actuation of the cursor. On the other hand, if the displayed screen is represented by less effective pixels in the system than in reality, interaction will be rather jaggy and imprecise. Latency and update rate can be improved by using a faster acquisition system.

In order for the system to work at its full capacity, the camera must be more sensible in the wavelengths range of the laser spot, otherwise, the identification of the laser spot could fail. Also, better contrast between the display image and the rest of the scene will result in better preprocessing, and therefore a better mapping.

More research is needed to formally compare the performance of an auto-calibrated laser pointer tracking system and a manually calibrated system of the same kind. This is a necessary step in assessing if an auto-calibrated pointing interface is a good alternative to manually calibrated systems.

Chapter 7

Conclusion

In this thesis, three optical trackers were implemented and studied: a 4 DOF pointing device, a 6 DOF mouse and a laser pointer based interaction device,.

The 4 DOF input device locates the user's hand position in three dimensions and also provides the *Roll* orientation component in its current configuration. It can also be configured to give another orientation component by changing the camera position. It provides data more accurately than the 6 DOF device but at the cost of precalculating a camera parameter (the *Field of View*) and the constraint of keeping the device always pointed towards the camera. It is a low cost, easy to use and not cumbersome, well adapted to use in virtual environments.

Left to do on this device is determining the optimal baseline between the two LEDs for more accurate positioning without disturbing user interaction. During the design of this device, we assumed correctly that hand width would not disturb the user and provide acceptable results. A larger baseline would necessarily give better results but how big can it be before it hinders user movement is an open research question, demanding a user study.

The 6 DOF mouse is a device that provides, through computation of a planar homography, the position and orientation of the user's hand in three dimensional space. Performance analysis shows that it could be a good input device for use in a virtual reality room, as long as accuracy requirements are not too high, especially concerning the orientation. On the other hand, it is a low-cost device that is not cumbersome and can be adapted to give a minimal footprint on the user's field of view.

To improve the performance and reduce noise in the measurements, future work on this device should include testing different filtering methods to stabilize measurements. A method for estimating radial distortion would also improve measurements' accuracy. Finally, the device needs a faster computing platform than the one used here in combination with the Vision Graph framework (see appendix A), in order to improve the update rate and reduce the latency.

The laser pointing interface auto-calibrates position and orientation of the camera relative to the screen to provide easy setup. We found that the use of an IR laser is preferable to a visible (red) laser because it makes small latency and inaccuracies imperceptible to the user. In addition, the use of an IR laser makes detection of the laser spot easier through the use of an interferential filter placed once the initial screen detection step is completed. That is because contrast is maximized for the particular wavelength of the laser, thus increasing overall robustness. Overall, this interface works reliably, provides good performance, is not cumbersome and is easy to use. Further research could look at adding a discrete input technique to allow selection and manipulation.

In conclusion, computer-vision based user interfaces are promising and in some cases,

already provide performance that challenges standard user interfaces. The use of the invisible (infrared) light spectrum makes those interfaces more transparent to the user and can also increase reliability of the system. It is important though to find cameras that detect IR correctly, which is not generally the case for the low-cost webcams. Secondly, all of these interfaces were developed and tested under a single camera setting. Using multiple cameras would probably increase range and accuracy, but also increase processing time. Finally, a formal comparison between manually calibrated and auto-calibrated methods is needed to quantify the performance difference between them. This is however out of the scope of this thesis and thus relegated to future work.

Appendix A

Vision Graph Framework

Computer vision is a wide field in computer science and the hardware tools to help along are numerous. Many companies provide cameras, capture boards and other interfaces. Almost all of them also provide also their own API to drive the hardware. With that, everyone has a different idea on how an image and a buffer should be implemented, how to initialize the hardware and process data. This leads to a total chaos when trying to integrate multiple devices or interchange them for an application.

In an attempt to make work easier with these tools, a system was built to conjugate all of these hardware pieces together and create a software environment that is easier to work with. The tool created is called the Vision Graph (VG). Graph because the whole architecture is based on the idea of having a graph of hardware in software.

A graph is basically a set of nodes connected together in a certain way. Two things distinguish a graph form another one: the way connections are made the de definition of each node. It shall be demonstrated that a system can be designed to represent any application of computer vision with a graph that can be built with the vision graph.

A.1 Active vs. Passive nodes

The first division between node types is in the purpose they serve. If a node is required to accomplish a task that it initiates by itself like grab an image or process it, that node is called an a *active nodes.* It owns a thread and runs on its own power. On the other hand, if the node's purpose is to serve other nodes, like for example a buffer that provides or stores data when asked, it is called a *passive node* and runs only when called on the caller's processing time slice.

A.2 The node manager

The node manager is a passive node that assures coherence in the graph. When a node is created, it is registered in the node manager, when deleted, it notifies the manager. This entity is in charge of starting and stopping execution of nodes in the graph as well as communication with the operating system.

A.3 Main nodes

Vision applications are created out of three main pieces. *Input, processing* and *output.* A node was created for each. Those main nodes are all *active nodes.* Each active node is driven by a thread that makes them run in parallel.

A.3.1 Grabber nodes

The first part of an application is data acquisition. In order to do any processing, there must be something to be processed. That is why the first node in this graph system is a

data capture node. It is named a *Grabber*. It is the abstraction of the input device used to acquire data for processing.

The *Grabber* can have only one connection in output because it can only provide one feed at a time.

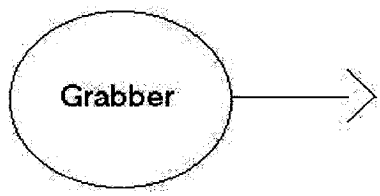

Figure A.1: The Grabber node.

A.3.2 Writer nodes

At the other end of the spectrum, the system must provide feedback to the outside world and therefore need output nodes that we called *Writer* nodes. These nodes are used to write data to an external interface. That is why they have only one input when it comes to graph connections. The output is outside the scope of the graph and is node dependant.

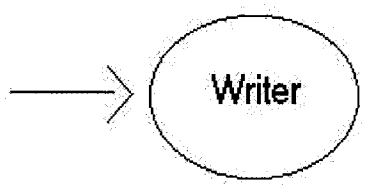

Figure A.2: The Writer node.

A.3.3 Processor nodes

The main goal of any computer vision application is to process incoming data, make something out of it and output a result. A third node was created for that purpose, the Processor node. This node takes one to N inputs and provides the same flexibility when it comes to output connections. The core of the application is to be built in these nodes.

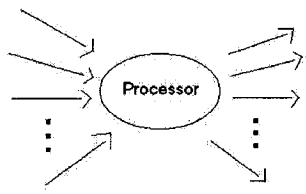

Figure A.3: The Processor node.

A.4 Buffer Nodes

In between nodes, information must be managed. Since every node is performing its task independently, data must be buffered intelligently to deal with speed discrepancies between nodes. The *Buffer* nodes were created to manage a set of memory buffers that will accomplish this task . Contrarily to other nodes seen until now, they are *passive nodes* meaning that they do not operate on their own, they are driven to action by other nodes. More technically speaking, the request sent to the buffer node will be processed in the stack of the thread that calls it. There are two types of *Buffers* nodes for two wide contexts of application.

A buffer node takes one input and provides N outputs (See figure A.4).

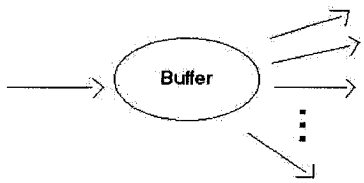

Figure A.4: The Buffer node.

A.4.1 Queue buffer

The *Queue buffer* node is the simplest one of the two. The set of memory buffer is managed as a simple queue of data. When a request to add data is made, this node allocates data and adds it to the back of a queue. When a request to obtain data is received, data is popped out of the front of the queue is made available as a pointer to the requesting node until released. When multiple nodes request data, two modes are available. The first mode pops information every time a request is made, the second mode waits for all nodes connected to process the same data before popping further data.

A.4.2 Quick buffer

The *Quick buffer* is a more complex structure that has the task to provide the latest information available to the requesting node.

Information submitted to this buffer is timestamped and stored in an internal double buffer. The use of a double buffer allows for one buffer to be always readable if a request for information comes in.

When a request for information is received, if the latest information is in the unused *double buffer*, that buffer is added to the *output buffer vector* and replaced in the *double*

buffer vector by a free buffer from the *free buffer queue.* A pointer to the information is sent back and references are incremented for that buffer.

If the latest information is in a buffer from the *output buffer vector*, a pointer to that buffer is sent back and references are incremented.

When a buffer is not referenced to, it is returned to the empty buffer queue unless it has the latest information.

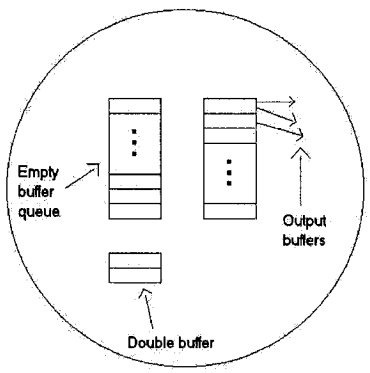

Figure A.5: The QuickBuffer node.

A.5 Other nodes

Other nodes can be added through deriving from one of the two main types of nodes. The only type added until now is the synchronizer node.

A.5.1 Synchronizer node

The synchronizer node acts like a street light to synchronize two or more nodes together. It waits for every node connected to it to report in before sending a signal back to all the connected nodes to tell them to proceed with the next iteration.

A.6 Examples of application

A simple application that can be trivially done with the VG is a viewer for the camera. It only requires a grabber and a writer. From the base Grabber class, we implemented a node for grabbing images for the camera and, from the Writer class, we implemented a node for writing images on the screen.

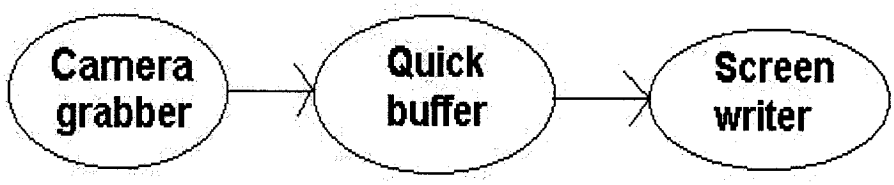

Figure A.6: Trivial VG application

A typical vision application will involve adding a Processor node in the middle along with another Quick buffer node before the writer.

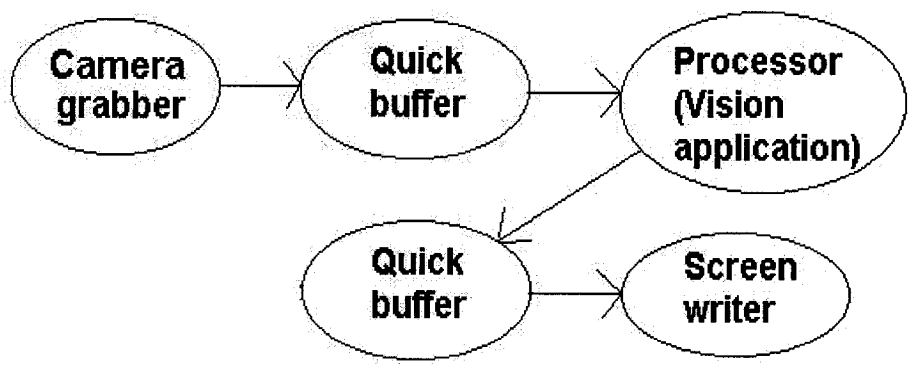

Figure A.7: Typical VG application

The VG has more capabilities that have not been exploited yet. For example an application could take input from two image sources, video, camera or others, and multicast them over a network socket writer on one machine. Another machine could grab from this stream, process and multicast on another channel, etc. Because of the parallel nature of the VG, such applications can be built with minimal effort.

Bibliography

[1] *Ascension technology corporation web site, http://www.ascension-tech.com.*

[2] *Fakespace website http://www.fakespace.com.*

[3] *Faro website http://www.faro.com.*

[4] *Intersense website http://www.isense.com.*

[5] *Logitech 3d mouse and head tracker technical reference manual.*

[6] *Polhemus web site, http://www.polhemus.com.*

[7] Bernard D. Adelstein, Eric R. Johnston, and Stephen R. Ellis, *Dynamic response of electromagnetic spatial displacement trackers*, Presence **5** (1996), no. 3, 302–318.

[8] Ronald T. Azuma, *A survey of augmented reality*, Presence: Teleoperators and Virtual Environments **6** (1997), no. 4, 331–356.

[9] R. Baribeau, G. Godin, L. Cournoyer, and M. Rioux, *Colour three-dimensional modelling of museum objects*, Proc. Imaging the Past Conference (London, U.K.), 1994, pp. 199–210.

[10] François Bérard, *The perceptual window: Head motion as a new input stream*, Proceedings of Human-Computer Interaction (INTERACT'99) (Edinburgh, UK), August 30-September 3 1999, pp. 238–244.

[11] F. Blais, J.-A. Beraldin, S.F. El-Hakim, and L. Cournoyer, *Real-time geometrical tracking and pose estimation using laser triangulation and photogrammetry*, Proceedings of Third International Conference on 3-D Digital Imaging and Modeling (Quebec, Canada), May 28 - June 1 2001, pp. 205–212.

[12] James Boritz and Kellogg S. Booth, *A study of interactive 6 dof docking in a computerised virtual environment*, Proceedings of IEEE Virtual Reality Annual International Symposium '98 (Atlanta, Georgia, USA), March 14-18 1998, pp. 139–146.

[13] Pierre Boulanger, Jean-François Lapointe, and Winston Wong, *Virtualized reality: an application to open-pit mine monitoring*, Proceedings of 19th International Society for

Photogrammetry and Remote Sensing (ISPRS) Congress (Amsterdam, The Netherlands), July 16-23 2000.

[14] Pierre Boulanger, John Taylor, Sabry El-Hakim, and Marc Rioux, *How to virtualize reality: An application to the re-creation of world heritage sites*, Proceedings of VSMM98 (Gifu, Japan) (International Society on Virtual Systems and Multimedia, eds.), vol. I, November 18-20 1998, pp. 39–45.

[15] Al Bovik (ed.), *Handbook of image & video processing*, Academic Press, San Diego, 2000.

[16] D. Bowman, D. Johnson, and L. Hodges, *Testbed evaluation of virtual environment interaction techniques*, Presence: Teleoperators and Virtual Environments **10** (2001), no. 1, 75–95.

[17] D. Bowman, D. Koller, and L. Hodges, *A methodology for the evaluation of travel techniques for immersive virtual environments*, Virtual Reality: Research, Development, and Applications **3** (1998), no. 2, 120–131.

[18] R. Brunelli and T. Poggio, *Template matching: matched spacial filters and beyond*, Tech. Report AIM-1549, IRST, 1995.

[19] Duncan Cavens, Florian Vogt, Sidney Fels, and Michael Meitner, *Interacting with the big screen: Pointers to ponder*, Proceedings of ACM Conference on Computer Human Interaction (CHI2002) (Minneapolis, Minnesota, USA), April 20-25 2002, pp. 33–40.

[20] J. Davis and X. Chen, *LumiPoint: Multi-user laser-based interaction on large tiled displays*, Displays **23** (2002), no. 5, 205–211.

[21] Richard R. Eckert and Jason A. Moore, *The classroom of the 21^{st} century: The interactive learning wall*, SIGCHI Bulletin **32** (2000), no. 2, 33–40.

[22] Olivier Faugeras, *Three-dimensional computer vision: A geometric viewpoint*, The MIT Press, Cambridge, Massachusetts, 1993.

[23] M.A. Fisher and R.C. Bolles, *Random sample consensus: A paradigm for model fitting with applications to image anlysis and tutomated cartography*, Comm. of the ACM **24** (1981), 381–395.

[24] Paul M. Fitts, *The information capacity of the human motor system in controlling amplitude of movement*, Journal of Experimental Psychology **47** (1954), 381–391.

[25] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes, *Computer graphics: Principles and practice*, second edition ed., Addison-Wesley, Reading, MA, 1990.

[26] Philippe Fuchs, Guillaume Moreau, and Jean-Paul Papin, *Le traité de la réalité virtuelle*, Les Presses de l'École des Mines de Paris, Paris, France, 2001.

[27] D.O. Gorodnichy, S. Malik, and G. Roth, *Nouse 'use your nose as a mouse' - a new technology for hands-free games and interfaces*, Proceedings of the 15th International Conference on Vision Interface (Calgary, Canada), May 2002, CD-ROM Only.

[28] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2000.

[29] Hewett, Baecker, Card, Carey, Gansen, Mantei, Perlman, Strong, and Verplank, *Acm sigchi curricula for human-computer interaction*, ACM SIGCHI, 1997.

[30] M. Ikits, J.D. Brederson, C. Hansen, and J. Hollerbach, *An improved calibration framework for electromagnetic tracking devices*, Procceedings of IEEE VR2001, 2001, pp. 63–70.

[31] T. Jebara and A. Pentland, *Parametrized structure from motion for 3d adaptive feedback tracking of faces*, Proceedings of Computer Vision and Pattern Recognition (CVPR), 1997.

[32] Dan R. Olsen Jr. and Travis Nielsen, *Laser pointer interaction*, Proceedings of SIGCHI'01 (Seattle, WA, USA), March 31 - April 4 2001, pp. 17–22.

[33] Volodymyr Kindratenko, *A comparison of accuracy of an electromagnetic and hybrid ultrasonic-inertia position tracking system*, Presence **10** (2001), no. 6, 657–663.

[34] Volodymyr V. Kindratenko, *A survey of electromagnetic position tracker calibration techniques*, Virtual Reality: Research, Development, and Applications **5** (2000), no. 3, 169–182.

[35] C. Kirstein and Heinrich Müller, *Interaction with a projection screen using a camera-tracked laser pointer*, Proceedings of International Conference on Multimedia Modeling (MMM 98) (Lausanne, Switzerland), October 12-15 1998, pp. 191–192.

[36] Reinhard Klette, Karsten Schlns, and Andreas Koschan, *Computer vision: Three-dimensional data from images*, Springer, Singapore, 1998.

[37] Senthil Kumar, *6d-mouse: A vision-based computer input device*, Tech. report, Bell Laboratories, 2000, http://www.bell-labs.com/user/senthil.

[38] Dominic Laberge, Jean-François Lapointe, and Emil M Petriu, *An auto-calibrated laser-pointing interface for large screen displays*, Proceedings of the International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2003) (Delft, The Netherlands), October 23-25 2003, *To appear.*

[39] Dominic Laberge, Jean-François Lapointe, Emil M Petriu, and Pierre Boulanger, *A 6-d mouse for virtual environments*, Proceedings of HAVE 2002 - IEEE International Workshop on Haptic Virtual Environments and their Applications (Ottawa, Ontario, Canada), November 17-18 2002, pp. 37–41.

[40] J.F. Lapointe, J.M. Robert, and P. Boulanger, *Optimizing performance in heavy equipment teleoperation*, Proceedings of Minespace 2001 (Quebec, Quebec, Canada), April 30 - May 3 2002.

[41] Shahzad Malik, *Robust registration of virtual objects for real-time augmented reality*, Master's thesis, Carleton University, 2002.

[42] Shahzad Malik, Gerhard Roth, and Chris McDonald, *Robust 2d tracking for real-time augmented reality*, Proceedings of the 15th International Conference on Vision Interface (Calgary, Canada), May 2002, CD-ROM Only.

[43] 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*, Proceedings of ACM Conference on Computer Human Interaction (CHI2002) (Minneapolis, Minnesota, USA), April 20-25 2002, pp. 33–40.

[44] Brad A. Myers, Choon Hong Peck, Jeffrey Nichols, Dave Kong, and Robert Miller, *Interacting at a distance using semantic snarfing*, Proceedings of Ubicomp 2001 (Atlanta, Georgia, USA), September 30 - October 2 2001, pp. 305–314.

[45] Harley R. Myler and Arthur R. Weeks, *The pocket handbook of image processing algorithms in c*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[46] N. Oliver, A. Pentland, and F. Berard, *Lafter: Lips and face real time tracker*, Proc. Computer Vision and Patt. Recog., 1997.

[47] Choon Hong Peck, *Useful parameters for the design of laser pointer interaction techniques*, Proceedings of ACM Conference on Computer Human Interaction (CHI2001) (Seattle, WA), 31 March-5 April 2001, pp. 461–462.

[48] W. H. Press, W. T. Vettering, S. A. Teukolsky, , and B. P. Flannery, *Numerical recipes in c++*, Cambridge University Press, Cambridge, 2002.

[49] F. Raab, E. Blood, T. Steiner, , and T. Jones, *Magnetic position and orientation tracking system*, IEEE Transactions on Aerospace and Electronic Systems **AES-15** (1979), no. 5, 709–718.

[50] J. Rehg and T. Kanade, *Digiteyes: Vision-based hand tracking for human-computer interaction*, Workshop on Motion of Non-Rigid and Articulated Bodies, November 1994, pp. 16–24.

[51] Yoichi Sato, Makiko Saito, and Hideki Koike, *Real-time input of 3d pose and gestures of a user's hand and its applications for hci*, 2000 IEEE International Conference on Automatic Face and Gesture Recognition (FG 2000), March 2000, pp. 462–467.

[52] Keiichi Sawada, Masayuki Okihara, and Shigeru Nakamura, *A wearable attitude-measurement system using a fiberoptic gyroscope*, Presence **11** (2002), no. 2, 109–118.

[53] Thomas B. Sheridan, *Musings on telepresence and virtual presence*, Presence: Teleoperators and Virtual Environments **1** (1992), no. 1, 120–126.

[54] Ben Shneiderman, *Designing the user interface: Strategies for effective human-computer interaction*, third edition ed., Addison-Wesley-Longman, Reading, MA, 1998.

[55] J. Sklansky, *Recognition of convex blobs*, Pattern Recognition **2** (1970), no. 1, 3–10.

[56] Chang Geun Song, No Jun Kwak, and Dong Hyun Jeong, *Developing an efficient technique of selection and manipulation in immersive v.e.*, Proceedings of the ACM symposium on Virtual reality software and technology (VRST'00) (Seoul, Korea), October 22-25 2000, pp. 142 – 146.

[57] Milan Sonka, Vaclav Hlavac, and Roger Boyle, *Image processing, analysis and machine vision*, second ed., PWS, Pacific Grove, 1998.

[58] Rahul Sukthankar, Robert G. Stockton, and Matthew D. Mullin, *Smarter presentation: Exploiting homography in camera-projector systems*, Proceedings of International Conference on Computer Vision (Vancouver, Canada), July 9-12 2001, pp. 247–253.

[59] K. Toyama, *Look, ma — no hands!' hands-free cursor control with real-time 3d face tracking*, Proc. Workshop on Perceptual User Interfaces (PUI'98) (San Francisco, CA), November 1998, pp. 9–54.

[60] B.A. Watson, N. Walker, W.R. Ribarsky, and V Spaulding, *The effect of variation in system responsiveness on user performance in virtual environments*, Human Factor **40** (1998), no. 3, 403–414, Special section on Virtual Environments.

[61] Du Wei and Li Hua, *Vision based gesture recognition using just one camera*, Proceedings of ICSP 2000 (Beijing, China), vol. 2, 2000, pp. 1351–1357.

[62] C.D. Wickens, *The effects of control dynamics on performance*, Handbook of Human Perception and Performance (New York) (K. Boff et al., ed.), vol. II, Wiley, 1986, pp. 39–1 to 39–60.

[63] T. Winograd and F. Guimbretiere, *Visual instruments for an interactive mural*, ACM SIGCHI CHI99 Extended Abstracts (Pittsburgh, PA), 1999, pp. 234–235.